



Executable Architecture of Net Enabled Operations: State Machine of Federated Nodes

M.G. Ball
R.W. Funk
DRDC CORA

Mr. R. Sorensen
Vitech Corp.

DRDC CORA TR 2009-012
November 2009

Defence R&D Canada
Centre for Operational Research & Analysis

Joint Staff Operational Research Team



National
Defence

Défense
nationale

Canada

Executable Architecture of Net Enabled Operations: State Machine of Federated Nodes

Mr. M.G. Ball
Mr. R.W. Funk
Joint Studies OR Team

Mr. R. Sorensen
Vitech Corp.

Defence R&D Canada – CORA

Technical Report
DRDC CORA TR 2009-012
November 2009

Principal Author

Original signed by M.G. Ball

M.G. Ball

Joint Studies OR Team

Approved by

Original signed by C.C. Morrissey

C.C. Morrissey

Acting Section Head Joint and Common OR

Approved for release by

Original signed by D. Reding

D. Reding

Chief Scientist

Defence R&D Canada – Centre for Operational Research and Analysis (CORA)

© Her Majesty the Queen in Right of Canada, as represented by the Minister of National Defence, 2009

© Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale, 2009

Abstract

The Defence Research and Development Canada (DRDC) Centre for Operational Research and Analysis (CORA) is developing capability-engineering analysis tools to support the building, demonstration, and analysis of executable architectures. Our previous paper [7] described how to model workflows within an Operations Centre (OPCEN) employing a Net-Centric architecture. It used a State Machine (SM) to simulate how multiple jobs can proceed in parallel, and in contention within priorities, when operators use a Task, Post, Process, Use (TPPU) cycle to organize their work.

This paper extends the OPCEN SM model to track the interaction of work between OPCENs. The State Machine of Federated Nodes (SMOFN) engine is organized around networked functional nodes within those OPCENs that produce and consume products held in a virtual Repository. The data-driven simulation uses files to build customized job workflows and configure any combination of nodes without affecting the operational logic.

Résumé

Le Centre d'analyse et de recherche opérationnelle (CARO) de Recherche et développement pour la défense Canada (RDDC) est en train de développer des outils d'analyse pour l'ingénierie des capacités, à l'appui du développement, de la démonstration et de l'analyse d'architectures exécutables. Dans notre document précédent *Erreur ! Source du renvoi introuvable.*, nous décrivions comment modéliser les flux de travaux dans un Centre des opérations (CENOP) utilisant une architecture réseaucentrique. Le modèle faisait appel à une machine à états pour simuler comment des tâches multiples peuvent être exécutées en parallèle, et en cas de conflit dans les priorités, lorsque les opérateurs utilisent un cycle TPPU (tâche, affichage, traitement et utilisation) pour organiser leur travail.

Dans le présent document, le modèle de machine à états des CENOP est élargi et permet de suivre l'interaction des tâches entre les CENOP. Le moteur de la machine à états de nœuds fédérés (SMOFN, de l'anglais State Machine of Federated Nodes) est articulé autour de nœuds fonctionnels en réseau dans ces CENOP, qui génèrent et consomment des produits dans un dépôt virtuel. La simulation dirigée par les données a recours à des fichiers pour créer des flux de travaux personnalisés et configurer une combinaison de nœuds sans nuire à la logique opérationnelle.

This page intentionally left blank.

Executive summary

Executable Architecture of Net Enabled Operations: State Machine of Federated Nodes:

M.G. Ball; R.W. Funk; R. Sorensen; DRDC CORA TR 2009-012; Defence R&D Canada – CORA; November 2009.

Background: The Defence Research and Development Canada (DRDC) Centre for Operational Research and Analysis (CORA) Joint Studies Operational Research Team (JSORT) has been engaged in several efforts to model processes within the realm of Command, Control, Computers, Communications, Intelligence, Surveillance and Reconnaissance (C4ISR). Through this work, it became apparent that the ability to interrupt jobs in favour of higher priorities created a unique challenge because it could not be modelled using classical sequenced activity diagrams. The chosen solution was to create a model of an operations centre (OPCEN) using the approach of a state machine (SM), which is defined in [4] as “any device that stores the status of something at a given time and can operate on input to change the status and/or cause an action or output to take place for any given change.” This led to the development of the OPCEN SM described in [5] through [8].

The SM of Federated Nodes (SMOFN) engine includes all of the functionality of the OPCEN SM and expands on it to account for interactions between several OPCENs operating collectively through a Net-Centric Architecture. It also adds the capability to simulate all aspects of OPCEN work beyond simply creating products. These expansions form the basis of the report.

Developing the SMOFN engine: The OPCEN SM has been extended to account for Net-Enabled interactions between several OPCENs. In essence, the OPCEN SM accounted for the work done by a single OPCEN to produce several products based on data analysis. In the SMOFN, not only are such production jobs tracked within several OPCENs, but the products they create are uploaded to a common, networked Repository so that all products can be accessed and used by any OPCEN. The SMOFN engine is an attempt to capture the essential logic that governs the way work is conducted anywhere, but with particular emphasis on ensuring that it can be fully applied to integrated activity between networked OPCENs.

Because the OPCEN SM concentrated on job production, most of the SMOFN development concentrated on adding the effect of nodes outside of the Producer, as well as creating the interfaces between all of the nodes. Including the Producer, there are five nodes, or types of activity, accounted for within the SMOFN, and each has a unique activity set which they are responsible for.

1. The **Producer** generates products from available data inputs;
2. The **Consumer** receives products which it uses to maintain situational awareness (SA). It may generate questions and send them to the **Discover** activity;

3. **Discovery** receives questions from the **Consumer** and searches to see if they can be answered using available information. Discovered information is passed to the **Producer** and missing information is requested from **External Sources**;
4. The **External Sources** activities receive queries and send new data to the **Producer**; and
5. The **Repository** stores information and acts as a medium for the other nodes to communicate.

Populating the SMOFN engine: The primary advantage of the SMOFN is that the model itself is only concerned with the execution of the logic. All operating parameters, such as the number of operators at an OPCEN and their skill sets, or the number of jobs to be done, are read from data files during the setup stage of the simulation. This data must be representative of the operating parameters of the modelled OPCENs, as the goal is to accurately simulate data flow between these. Particular emphasis is placed on describing job threads that involve multiple OPCENs, such as the process to report and respond to events in theatre or the preparation of high-level daily briefs. Another major component is the composition of various OPCENs, as far as the number of operators on shift and the skills they are trained in. Although much of this data will be classified, the SMOFN engine itself remains unclassified as it is saved in a separate file from any data which it would read upon execution.

This work starts with the analysts converting whatever evidence exists (typically documentation or verbal descriptions from operators) of the current Command and Control (C2) practices into model form. In theory these should be Standard Operating Procedures (SOPs) that execute as a thread from start to finish. The actual experience is that most SOPs tend to be incomplete or inaccurate so their logic will not actually execute as described. The knowledge gained through previous work with the SMOFN engine is used to address these gaps so that the modelling process eventually captures fully executable threads. Not only can these threads be recreated as input files to expand the scope of the SMOFN, but if necessary, the execution logic can be enhanced by incorporating new operational rules that the analysts become aware of through the modelling process. The resulting threads can then be used to articulate the C2 processes as refined SOPs and the model is available for further analysis and improvement.

A major discussion item that came out of one of the SMOFN development workshops was the need to see how the SOP modelling process actually works. The consensus was that it would be easier to understand if one or two examples could be modelled during the workshop. The workshop participants selected two candidates of important C2 practices; one had no organized SOP while the other SOP was considered to be the most complex one that had been documented. The thought was that if these two cases could be successfully modelled it would provide ample evidence as to the viability of the modelling approach.

In each case the approach taken was to capture information about the job thread within CORE (a software package designed for system engineering and also used for process modelling) and then read through the output transcripts with the operator Subject Matter Expert (SME) to make sure the model executed as intended.

Results: The SMOFN has achieved its goal of extending the detailed OPCEN SM model logic across multiple nodes, each with their own jobs and resources but also with the ability to share information. This information sharing takes place through the Net Centric virtual Repository,

whose operational logic successfully controls the interaction of information between operator perspectives (i.e. producer, consumer and discovery) as well as between different OPCENs within the SMOFN.

Typical simulations are limited in scope by only being able to model what is built directly into them. On the other hand, SMOFN reads scenario information from data files each time the simulation is started. This allows a single version of the model to be used across a limitless variety of scenarios. It also allows any details of the SMOFN engine to remain unclassified, even though the files that it uses at input may contain classified data.

Both SOP modelling trial cases successfully demonstrated how to quickly (a few days or weeks from start to finish) build and validate detailed models with operator SMEs. The results were much better articulated C2 processes than existed before and they provided a means to test a broad range of process options. The added bonus was that formatted United States Department of Defense Architecture Framework (DoDAF) or Canadian Department of National Defence / Canadian Forces Architecture Framework (DND AF) product documents could be automatically generated for whatever data was entered into the model. This gives the SMEs something to take back as a documented SOP.

Future plans: The process of gathering data for process threads is both quick and effective; the major limitation noted so far is the availability of SMEs to provide the operational context. This is expected to continue to be the biggest implementation hurdle to building a faithful simulation and executable Canadian Forces (CF) Command Structure operational architecture. What could convince operators to make this time investment is the fact that the SME interaction produces immediate results in the form of DoDAF/DND AF-compatible products that can be used by the SMEs' home organizations as the basis for writing proper SOPs. There are several SOPs that have been suggested as future candidates to be included in the SMOFN, on the grounds that the SMEs believed those processes could be improved if they were properly articulated and documented. These would be given the same treatment as the trial cases modelled here.

The final step in the modelling will involve converting the C2 process threads to SMOFN data input file format. A portion of this has already been demonstrated using the above trial cases by decomposing the threads into simple sub-jobs and then linking them together.

Significance: The SMOFN engine is a viable simulation of OPCEN operations and interactions. Although a large amount of data will be required to make the model reflective of real processes, the authors' recommendation is that the SMOFN be used as a testbed for potential changes to operational procedures. To make the simulation and its results as realistic as possible, it is also recommended that the necessary SME hours be made available such that the full CF Command Structure could be modelled to some degree of fidelity. This investment of time would make SMOFN a valuable tool in studying how the CF does, and potentially could, operate.

Sommaire

Executable Architecture of Net Enabled Operations: State Machine of Federated Nodes:

M.G. Ball; R.W. Funk; R. Sorensen; DRDC CORA TR 2009-012; R & D pour la défense Canada – CORA; Novembre 2009.

Contexte : L'Équipe de recherche opérationnelle interarmées (ERO (IA) du Centre d'analyse et de recherche opérationnelle (CARO) de Recherche et développement pour la défense Canada (RDDC) a déployé de multiples efforts pour modéliser les processus reliés au champ professionnel Commandement, contrôle, communications, informatique, information, surveillance et reconnaissance (C4ISR). Dans le cadre de ces travaux, il est devenu évident que la capacité d'interrompre des tâches au profit d'autres tâches, celles-là plus prioritaires, allait poser un défi unique puisque cette capacité était impossible à modéliser au moyen des graphiques d'activités ordonnées classiques. La solution retenue a été de créer un modèle pour un centre d'opérations (CENOP), conformément à l'approche d'une machine à états qui est définie comme [4] étant « un appareil permettant de stocker l'état d'une chose à un moment donné et pouvant agir sur les entrées afin de modifier l'état et/ou déclencher une action ou la production d'une sortie pour un changement donné ». Cela a mené au développement de la machine à états pour les CENOP qui est décrite aux points [5] à [8].

Le moteur de la machine à états de nœuds fédérés comporte toutes les fonctionnalités de la machine à états pour les CENOP, en plus d'englober les interactions entre plusieurs CENOP exploités collectivement par le biais d'une architecture réseaucentrique. Il permet également de simuler tous les aspects du travail d'un CENOP au-delà de la simple création de produits. Ce sont ces éléments ajoutés qui constituent la base de ce rapport.

Développement du moteur de la machine à états de nœuds fédérés : La machine à états pour les CENOP a été élargie de manière à ce que les interactions réseaucentriques entre plusieurs CENOP soient prises en compte. Essentiellement, la machine à états pour les CENOP permettait de tenir compte du travail exécuté par un seul CENOP et de générer ainsi plusieurs produits, d'après des analyses de données. Dans la machine à états de nœuds fédérés, ce genre de tâches de production fait non seulement l'objet d'un suivi dans plusieurs CENOP, mais tous les produits que ces centres créent sont aussi téléversés dans un dépôt en réseau commun afin qu'ils puissent être accessibles et utilisables par tous les CENOP. Avec le moteur de la machine à états de nœuds fédérés, on tente de représenter la logique essentielle régissant la façon dont le travail est accompli partout, mais en insistant particulièrement sur la nécessité de s'assurer que ce moteur pourra s'appliquer intégralement aux activités intégrées entre les CENOP en réseau.

Étant donné que la machine à états pour les CENOP privilégiait la production de travail, la plupart des activités de développement de la machine à états de nœuds fédérés ont surtout consisté à ajouter l'effet des nœuds à l'extérieur du Producteur de même qu'à créer les interfaces entre tous les nœuds. Producteur y compris, il y a cinq nœuds, ou types d'activités, de représentés dans la machine à états de nœuds fédérés, et chacun de ces nœuds comporte un ensemble d'activités uniques dont il est responsable.

6. Le **Producteur** génère des produits à partir des entrées de données disponibles.
7. Le **Consommateur** reçoit les produits dont il se sert pour le maintien de la connaissance de la situation (CS). Il peut générer des questions et les envoyer à l'activité **Découverte**.
8. **Découverte** reçoit les questions du **Consommateur** et cherche pour déterminer s'il est possible d'y répondre à partir de l'information disponible. L'information découverte est transmise au **Producteur** et celle qui manque est demandée à des **Sources de l'extérieur**.
9. Les **Sources de l'extérieur** reçoivent les demandes d'information et envoient les données manquantes au **Producteur**.
10. Le **Dépôt** enregistre l'information et permet aux autres nœuds de communiquer.

Alimentation du moteur de la machine à états de nœuds fédérés (SMOFN) : Le principal avantage de cette machine à états réside dans le fait que le modèle à proprement parler se rapporte uniquement à l'exécution de la logique. Tous les paramètres d'exploitation, tels que le nombre d'opérateurs dans un CENOP et leurs compétences, ou le nombre de tâches à accomplir, sont extraits de fichiers de données à l'étape de la configuration de la simulation. Ces données doivent être représentatives des paramètres d'exploitation des CENOP modélisés, l'objectif étant de simuler avec exactitude le flux de données entre ces derniers. L'accent est mis en particulier sur la description des fils de tâches dans lesquels interviennent de multiples CENOP, comme le processus servant à signaler les événements dans un théâtre d'opérations et à y réagir, ou la préparation de comptes rendus généraux quotidiens. Une autre composante importante est la composition des divers CENOP, en ce qui concerne le nombre d'opérateurs pour un poste en particulier ainsi que les compétences pour lesquelles ils ont été formés. Malgré le fait que la majorité de ces données sera classifiée, le moteur de la machine à états de nœuds fédérés, lui, demeurera non classifié étant donné qu'il sera enregistré dans un fichier différent de celui qui renfermera les données qu'il lira lorsqu'il sera exécuté.

Les analystes commencent par convertir, en un modèle, les éléments d'information disponibles (en général, de la documentation ou des descriptions verbales faites par les opérateurs) sur les pratiques relatives au commandement et au contrôle (C2). En théorie, ces pratiques devraient être sous forme d'Instructions permanentes d'opération (IPO) s'exécutant comme un fil du début à la fin. Or, la vérité est que la plupart des IPO sont généralement incomplètes ou inexactes, de sorte que, dans les faits, leur déroulement logique diffère de la description qui en est faite. Les connaissances acquises dans le cadre des travaux antérieurs portant sur le moteur de la machine à états de nœuds fédérés servent à combler ces écarts de façon à faire en sorte que le processus de modélisation finisse par représenter des fils entièrement exécutables. Les fils peuvent non seulement être recréés comme des fichiers d'entrée pour élargir la portée de la machine à états de nœuds fédérés, mais, s'il y a lieu, la logique d'exécution peut être améliorée par l'ajout de nouvelles règles opérationnelles que les analystes découvrent par le biais du processus de modélisation. Les fils ainsi obtenus peuvent ensuite être utilisés pour organiser les processus C2 puisqu'il existe des IPO bien élaborées et un modèle permettant de faire d'autres analyses et améliorations.

Il est ressorti d'un des ateliers sur le développement d'une machine à états de nœuds fédérés une question importante et c'est la nécessité de voir comment le processus de modélisation des IPO

fonctionne vraiment. Il y avait unanimité que ce processus serait plus facile à comprendre s'il était possible de présenter un ou deux modèles durant l'atelier. Les participants de cet atelier ont sélectionné deux cas représentatifs de pratiques C2 importantes : dans le premier cas, il n'y avait pas d'IPO structurée et, dans le second, l'IPO était considérée comme étant l'une des plus complexes qui aient été documentées. De l'avis général, la modélisation de ces deux cas, si elle était réussie, permettrait de prouver amplement la viabilité de l'approche de modélisation.

Pour chacun des deux cas, l'approche retenue a consisté à représenter l'information sur le fil des tâches dans le logiciel CORE (un logiciel conçu pour la systémique et aussi utilisé pour la modélisation des processus), puis à lire les transcriptions obtenues avec l'expert en matière d'opérateurs (EEM) afin de s'assurer que le modèle s'exécutait comme prévu.

Résultats : La machine à états de nœuds fédérés a permis d'atteindre le but qui était d'étendre la logique détaillée du modèle de la machine à états pour les CENOP à des nœuds multiples, chaque nœud ayant ses propres tâches et ressources, tout en permettant le partage de l'information. Ce partage de l'information est rendu possible grâce au dépôt réseautique virtuel dont la logique opérationnelle permet de contrôler efficacement l'interaction de l'information entre les perspectives des opérateurs (c.-à-d. le Producteur, le Consommateur et l'activité Découverte) de même qu'entre les différents CENOP dans la machine à états de nœuds fédérés.

Les simulations typiques ont une portée limitée, car elles ne permettent que de modéliser ce qui y est intégré directement. Par ailleurs, la machine à états de nœuds fédérés extrait l'information des scénarios contenus des fichiers de données chaque fois que la simulation est lancée, ce qui permet d'utiliser une seule version du modèle pour un éventail infini de scénarios et de faire en sorte que tous les détails du moteur de la machine à états de nœuds fédérés restent non classifiés même si les fichiers auxquels ce moteur a recours renferment des données classifiées.

Les deux cas de modélisation des IPO testés ont permis de démontrer avec succès comment créer et valider rapidement (en seulement quelques jours ou semaines, du début à la fin) des modèles détaillés avec des opérateurs experts en la matière. Les processus C2 ainsi obtenus étaient beaucoup plus structurés que ceux qui existaient avant et ils permettaient de tester un vaste éventail de processus possibles. En plus, il était possible de générer automatiquement des documents de produits dans un format compatible avec le United States Department of Defense Architecture Framework (DoDAF) ou le cadre d'architecture du ministère de la Défense nationale et des Forces canadiennes (DNDAF) pour toutes les données incorporées dans le modèle, des documents que les EEM pouvaient soumettre comme IPO documentées.

Plans futurs : Le processus de collecte des données pour les fils des processus est à la fois rapide et efficace; la principale limite observée à ce jour est le fait qu'il n'y a pas d'EEM disponibles pour fournir le contexte opérationnel. Sur le plan de la mise en œuvre, cela devrait continuer d'être le plus gros obstacle au développement d'une architecture opérationnelle simulée et exécutable qui soit fidèle pour la structure de commandement des Forces canadiennes (FC). Ce qui pourrait convaincre les opérateurs de faire cet investissement en temps est le fait que l'interaction des EEM donne des résultats immédiats sous la forme de produits compatibles avec les DoDAF/DNDAF, qui peuvent servir de base, aux organisations d'attache de ces experts, pour la rédaction de leurs propres IPO. Les EEM ont suggéré que plusieurs IPO soient incorporées dans la machine à états de nœuds fédérés parce qu'ils croient que ces processus pourraient être

améliorés s'ils étaient bien structurés et documentés. Ces IPO feraient l'objet du même traitement que les cas d'essai modélisés dont il est ici question.

L'étape finale de la modélisation consistera à convertir les fils de processus C2 dans un format de fichier d'entrée de données pour la machine à états de nœuds fédérés. On a déjà fait la démonstration d'une partie de cette étape dans le cadre de la modélisation des cas d'essai susmentionnés, en décomposant les fils en sous-tâches simples, puis en les liant entre elles.

Portée : Le moteur de la machine à états de nœuds fédérés est une simulation viable des opérations des CENOP et de leurs interactions. Malgré le fait qu'il faudra une grande quantité de données pour faire en sorte que le modèle reflète les processus véritables, les auteurs recommandent que la machine à états de nœuds fédérés servent de banc d'essai pour tester des changements éventuels dans les procédures opérationnelles. Pour rendre la simulation et ses résultats le plus réalistes possible, on recommande aussi que les heures-personnes EEM nécessaires soient accordées pour permettre la modélisation assez fidèle de toute la structure de commandement des Forces canadiennes. Le temps qui serait ainsi investi par ces EEM permettrait de faire de la machine à états de nœuds fédérés un outil très utile pour étudier comment les FC fonctionnent à l'heure actuelle et comment elles pourraient éventuellement fonctionner.

This page intentionally left blank.

Table of contents

Abstract	i
Résumé	i
Executive summary	iii
Sommaire	vi
Table of contents	xi
List of figures	xiv
List of tables	xvii
Acknowledgements	xviii
1 Introduction.....	1
1.1 Background	1
1.2 Primer on TPED vs. TPPU	2
1.3 Overview of OPCEN SM Model.....	3
1.4 SMOFN Contract Summary	6
1.5 Aim.....	6
1.6 Scope of the Modelling Effort.....	7
1.7 Layout of Report.....	7
2 SMOFN Engine Modelling Concepts.....	9
2.1 Extension Beyond the OPCEN SM.....	9
2.2 Conceptual OV-1	9
2.3 Examples of the Process	11
2.4 Logic Within Each Node	12
2.4.1 Producer Logic	12
2.4.2 Consumer Logic	12
2.4.3 Discover logic	13
2.4.4 External Sources Logic	13
2.4.5 Repository Logic.....	13
2.5 Producer-Repository-Consumer Model.....	14
3 SMOFN Engine in CORE.....	15
3.1 SMOFN Top Level Description	15
3.2 Logic Within Branches.....	17
3.2.1 Main Branch.....	17
3.2.2 Clock Branch.....	17
3.2.3 Consume Branch	17
3.2.4 Produce and Discover Branch	17
3.2.5 Repository Branch.....	18
3.2.6 External Sources Branch	18

4	SMOFN Input Data Files	19
4.1	Simulation Setup	19
4.1.1	Switchboard	19
4.2	OPCEN Inputs	20
4.2.1	Thread Types.....	20
4.2.2	Thread Definitions	21
4.2.3	Events List.....	23
4.2.4	Operator Skills	23
4.2.5	Utility Decay Curves.....	23
4.3	Repository Inputs.....	24
4.3.1	Data Arrival Schedule	24
4.3.2	New Product Arrival Schedule.....	25
4.3.3	Delivery Matrix.....	25
4.3.4	Bandwidth	26
4.3.5	Question Generation.....	26
4.3.6	Discovery Requests.....	26
4.3.7	External Response.....	26
5	Data Collection via Process Modelling.....	28
5.1	Description of Proposed Method.....	28
6	Trial of SOP Modelling Process	30
6.1	Case 1 – eBrief	30
6.1.1	Context Leading to Approach	30
6.1.2	Description of the Modelled eBrief Process	31
6.2	Case 2 – Casualty Reporting	34
6.2.1	Context Leading to Approach	34
6.2.2	Description of the Modelled Casualty Reporting Process	36
6.3	Findings of the Trial	41
6.4	Description of Proposed Approach.....	42
7	Conclusions and Recommendations	43
	References	44
	Annex A .. SMOFN User Manual.....	47
A.1	Setup.....	47
A.1.1	Install the scripts:	47
A.1.2	Install Initialization Files:	47
A.1.3	Load the SMOFN project in CORE:	47
A.2	Simulation Procedure	48
A.2.1	Scenario Development – Scenario General Parameters	49
A.2.2	Scenario Development – Operations Centers.....	53
A.2.3	Scenario Development – Repository.....	61
A.2.4	Execution	68

A.2.5 Results	73
Annex B .. SMOFN Diagrams	79
Annex C .. Case 1 – eBrief	92
Annex D .. Case 2 – Casualty Reporting	104
Annex E... Future SOP Considerations	117
E.1 SME Proposed SOP Threads.....	117
E.2 SJS NDCC Proposed SOPs	117
E.3 JIIFC Det proposed SOPs.....	118
List of symbols/abbreviations/acronyms/initialisms	120
Glossary	122
Distribution list.....	123

List of figures

Figure 1: Task, Post, Process, Use (TPPU)	3
Figure 2: OPCEN SM Top Level	5
Figure 3: Scale-free SMOFN Operational Concept Graphic OV-1	10
Figure 4: Producer-Repository-Consumer Paths	14
Figure 5: SMOFN Top-level Model	16
Figure 6: Planned Operational Architecture Creation Process	28
Figure 7: Top Level Daily SJS NDCC eBrief	32
Figure 8: Final Review and Approval Process for SJS NDCC eBrief	33
Figure 9: International Casualty Reporting Model Implied by the SOP Matrix	35
Figure 10: Top Level International Casualty Reporting Model (SME Version)	37
Figure 11: International Casualty Reporting Model Showing Unit Interactions in Serials 1-8.....	38
Figure 12: Casualty Reporting Within The Internationally Deployed Unit During Serials 1-8....	40
Figure 13: Folder structure example.....	50
Figure 14: Simulation Start Time Example	50
Figure 15: S4-Switchboard Example.....	51
Figure 16: S5-OpCenters.txt Example.....	51
Figure 17: OPCEN 1 Folder Contents	52
Figure 18: Setting the Snapshot Interval	53
Figure 19: O1-ThreadTypes Example	55
Figure 20: O2-ThreadDefinitions Example	56
Figure 21: O2-ThreadDefinitions with Discovery Thread Highlighted	57
Figure 22: O3-Events Example	58
Figure 23: O4-SkillsMatrix Example	59
Figure 24: O5-DecayCurves Example.....	61
Figure 25: R1-ExternalData example	62
Figure 26: R2-ExternalProducts Example.....	63
Figure 27: R3-Delivery Example	64
Figure 28: R4-Bandwidth Example.....	65
Figure 29: R5-Questions Example	66
Figure 30: R6-DiscoveryRequests Example.....	67

Figure 31: R7-ExternalResponse Example.....	68
Figure 32: State Machine Element in CORE.....	69
Figure 33: Simulator Control Panel.....	70
Figure 34: Simulation Output Transcript.....	71
Figure 35: Simulation Complete Dialog.....	72
Figure 36: Model Changes Dialog Box.....	72
Figure 37: Results1-Init.csv Example	73
Figure 38: Results2-Operators.csv Example	74
Figure 39: Results3-Events.csv example.....	75
Figure 40: Results4-Events.Med.csv Example.....	75
Figure 41: Results5-Events.Short.csv Example	76
Figure 42: Results6-Full.Aggregation.csv Example.....	77
Figure 43: Results7-Partial.Aggregation.csv Example.....	78
Figure 44: SM State Machine EFFBD	80
Figure 45: Simulation Control.....	81
Figure 46: OPCEN Input.....	82
Figure 47: OPCEN Activity	83
Figure 48: OPCEN Output and Simulation Output.....	84
Figure 49: SM.1 Setup.MultiNode EFFBD.....	85
Figure 50: SM.12 Thread & Queue Processing EFFBD	86
Figure 51: SM.12.7 Job Queue Processing Branches EFFBD	87
Figure 52: SM.17 Send from Repository EFFBD	88
Figure 53: SM.18 Receive at Repository EFFBD	89
Figure 54: SM.22 End-of-Cycle Reporting EFFBD.....	90
Figure 55: SM.23 End-of-Run Reporting EFFBD	91
Figure 56: DB Daily Brief EFFBD	92
Figure 57: DB.1 NLT 2300 SWO & OWO select Ops items EFFBD	93
Figure 58: DB.2 Int/CDI Strat Inputs EFFBD.....	93
Figure 59: DB.3 Build E-Brief EFFBD.....	94
Figure 60: DB.3.11 0100-0130 Build high level summary slides EFFBD.....	95
Figure 61: DB.3.13 Publish high level summary to CV EFFBD	96
Figure 62: DB.4 Ops SITREPS EFFBD.....	97
Figure 63: DB.5 Briefing Approval EFFBD	97

Figure 64: DB.6 CDS Briefing EFFBD.....	99
Figure 65: DB.7 Other E-Portions EFFBD	100
Figure 66: DB.7.2 Key Issues & Support highlights Checks EFFBD.....	101
Figure 67: DB.7.3 CF personnel deployed inputs check EFFBD.....	102
Figure 68: DB.8 Briefing Dissemination EFFBD	102
Figure 69: DB.9 2000-0800 EFFBD	103
Figure 70: IP Action in the Event of Death/Serious Injury - International Ops - Parallel Processing EFFBD	104
Figure 71: IP.1 SER 1-8 - HR or Cas EFFBD	105
Figure 72: IP.1.1: Deployed Ops Activities EFFBD	106
Figure 73: IP.1.2 CEFCOM Activities EFFBD.....	107
Figure 74: IS.4.6 conduct IOPG and follow-ups as required EFFBD	108
Figure 75: IP.1.3 NDCC Activities EFFBD	108
Figure 76: IP.1.4: SJS Activities EFFBD	109
Figure 77: IP.1.5 CANOSCOM Activities EFFBD	109
Figure 78: IP.1.6 COMDS / ECS / L1 Activities EFFBD	110
Figure 79: IP.1.7 NMR Activities EFFBD	111
Figure 80: IP.3 SER 9 - HR EFFBD	112
Figure 81: IP.5 SER 10 - HR EFFBD	113
Figure 82: IP.6 SER 9 - Cas EFFBD.....	114
Figure 83: IP.8 SER 10 - Cas EFFBD.....	115
Figure 84: IP.9 SER 11-12 - HR or Cas EFFBD.....	116

List of tables

Table 1: Examples of the Process.....	11
Table 2: Distribution of Unit Activities per Serial as Listed in SOP Matrix.....	34
Table 3: Comparison of SOP Matrix and SME Modelled Activities & Triggers.....	36

Acknowledgements

Discussions with Pete Smith and Neil Carson of MARLANT N02OR as well as Maj. Art Henry and LCdr. Rob Peck of JIIFC Project were essential to the development of the operational logic pertaining to interactions between nodes.

The Daily eBrief and Casualty Reporting models were made possible through interviews with NDCC officers LCdr. Gavin McCallum and Lt(N). Steve Chouinard, respectively.

1 Introduction

The State Machine of Federated Nodes¹ (SMOFN) has been created as an engine to drive simulations of work performed by people who are able to share information over a common network. We refer to it as an “engine” because it contains the logic to handle the modelled concepts in a generic sense, while the specific details of the simulation are read upon execution. In this sense, the SMOFN becomes a “model” when it’s inherent coding and the necessary data files are considered together. Its generic logic allows any number of specific real world scenarios, either current or potential, to be studied by tailoring the input data to each situation. While the SMOFN logical rules are generic enough to allow for various types of work, they were inspired by data analysis tasks performed at an operations centre (OPCEN²) for the purpose of providing information to the OPCEN commander, or to other OPCENs. The SMOFN expands on previous work, described below, by allowing for collaboration between multiple nodes interacting over a shared network. It also adds the concept of follow-up work, such as asking questions and requesting new data, based on what has already been completed.

1.1 Background

In 2003, the Defence Research and Development Canada (DRDC) Collaborative Capability Definition, Engineering and Management (CapDEM) Technology Demonstration (TD) project began development of system engineering processes that were organized around and leveraged the United States Department of Defense Architecture Framework (DoDAF) [2]. To study and evaluate these processes, CapDEM required an organization or system to form the basis of an executable architecture model (an operational and/or system architecture model that can be run, or executed, as a simulation in order to validate the modelled logic or study the emerging behaviour), as well as a software package to capture the data in and aid in the analysis. The Joint Intelligence and Information Fusion Capability (JIIFC) project was chosen as a convenient use-case and CORE, by Vitech Corporation, was chosen as the software application.

Meanwhile, the DRDC Centre for Operational Research and Analysis (CORA) Joint Studies Operational Research Team (JSORT) had been assigned the responsibility of providing analysis support to JIIFC and was looking for appropriate modelling and simulation (M&S) methods and tools to support that activity. CapDEM and JSORT agreed that long-term research collaboration would help to improve the Department’s ability to model complex command and control (C2) architectures.

Another of JSORT’s research thrusts was in direct support of the Canadian Forces (CF) Command, Control, Computers, Communications, Intelligence, Surveillance and Reconnaissance (C4ISR) Campaign Plan (CP) [3]. This effort included the development of an analytical basis to

¹ The term “node” is used as an abstraction of the different roles performed by an OPCEN (Producer, Consumer, etc.), of the OPCENs themselves, or of the different roles within the chain of command. These are “federated” in the sense that they are operating as one large interconnected entity – their relationship is the operational equivalent of a federated database [1].

² Although the modelling was based on an “operations” centre, and the term OPCEN is used throughout this paper, the resulting logic applies equally to any type of command or support centre that performs identifiable tasks.

allow capability-engineering staff to build and assess executable Net-Centric architectures (i.e., executable architectures that are based upon, and take advantage of, a networked environment). The objective was to define requirements for future capabilities and make measurable investment decisions by helping operators to build realistic C2 architectures that capability engineers can use to compare current and planned options.

During several modelling efforts, it became apparent that the ability to interrupt jobs in favour of higher priorities created a unique challenge because it could not be modelled using classical sequenced activity diagrams. This was a significant concern because the ability to interrupt jobs is an important difference between the current job processing paradigm - Task, Post, Process, Use (TPPU) - and its predecessor Task, Process, Exploit, Disseminate (TPED). The chosen solution was to create a model of an OPCEN using the approach of a state machine (SM), which is defined in [4] as “any device that stores the status of something at a given time and can operate on input to change the status and/or cause an action or output to take place for any given change.” This led to the development of the OPCEN SM described in [5] through [8]. This was a leap forward in process modelling, especially where human behaviour and decision making are involved because it provides a structured approach to explicitly handle contention considerations when jobs must be interrupted for higher priority work and then returned to later. A short summary of that work is covered in the next two sections.

1.2 Primer on TPED vs. TPPU

TPED logic implies that each job is a serial process: jobs are worked on from start to finish without being interrupted. Flowchart diagrams (ie. those in CORE and similar modelling packages) can simulate TPED by putting the job processes in a loop and repeating for each job. TPPU, on the other hand, allows jobs to be interrupted by higher priority jobs. As illustrated in Figure 1, updates of jobs processed through TPPU are posted at regular intervals and the utility of the job increases as more posts are made. This accrued utility is saved even if the job is interrupted. TPPU leads to concurrent behaviour (several partially processed jobs existing at the same time) that is too complex to model using classical flowchart diagrams. The OPCEN SM overcame this by stopping time and executing the logic to reassign operators to jobs in order of priority. It then steps forward in time and repeats the process. Progress within each job is tracked by recording the job's state and updating it at each time step.

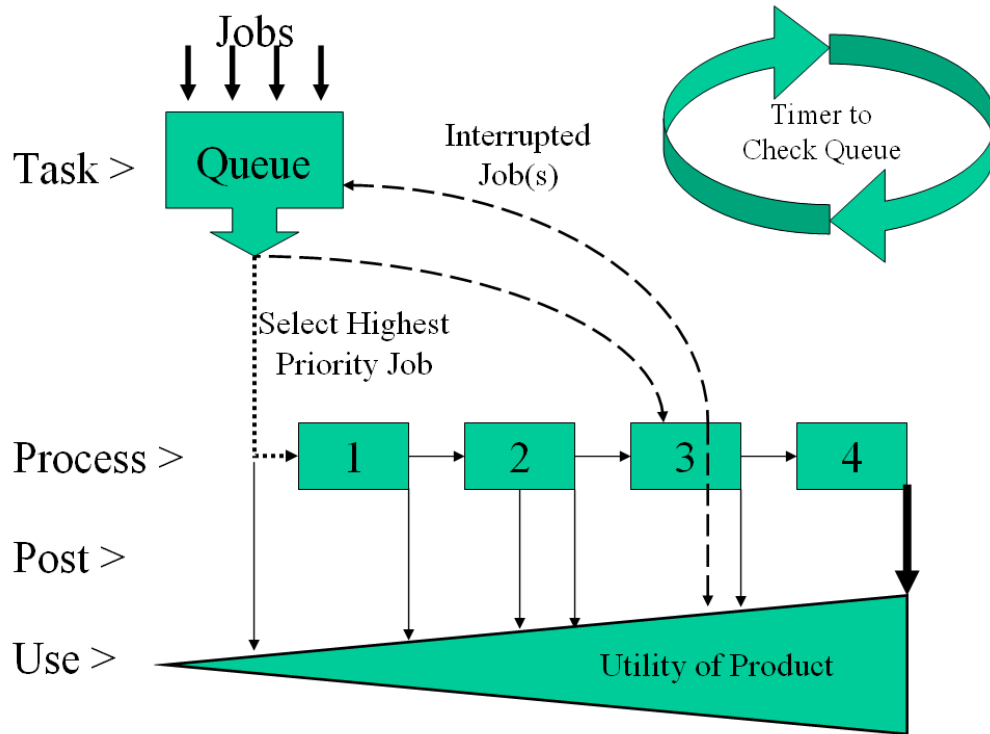


Figure 1: Task, Post, Process, Use (TPPU)

1.3 Overview of OPCEN SM Model

The OPCEN SM logic flow is illustrated in Figure 2. The actual simulation is contained inside the red box, whereas the rest of the diagram represents activities performed to control the input to and output from the simulation. Examining the diagram from left to right, the following activities are executed:

1. **Setup** reads data from input files and saves the data to variables stored in memory to be accessed by the simulation;
2. The simulation enters a Loop (**LP**) which will be executed for every time step. This loop includes steps 3 through 7 below;
3. **Run-to-End-Check** reads the clock variable to see if the time step designated as the end of the simulation has been reached. If so, the loop exits, otherwise the simulation enters the red box;
4. **Clock** increments the clock variable by one time step and sends the signal to the bottom branch indicating that it can now execute;
5. **Schedule Processing** receives that signal and reads the variables from the input data to see if any jobs are scheduled to begin during this time step. If so, those jobs are added to the events list;

6. **Thread & Queue Processing** performs the actual simulation of activity that occurs during the current time step. The details of how this is accomplished are handled at a lower level of decomposition;
7. **End-of-Cycle Reporting** saves to output variables any data that needs to be accessed by later time steps or output at the end of the simulation;
8. Once the Run-to-End-Check finds that the simulation has reached the final time step and the Loop finishes, **End-of-Run Reporting** goes through the data that has been saved to variables, and outputs that data as a series of spreadsheets.

The SMOFN engine includes all of the above functionality and expands on it to account for interactions between several OPCENs operating collectively through a Net-Centric Architecture. It also adds the capability to simulate all aspects of OPCEN work beyond simply creating products, such as posing questions based on newly received products, and searching for older products to fill information gaps. These expansions form the basis of the rest of this report.

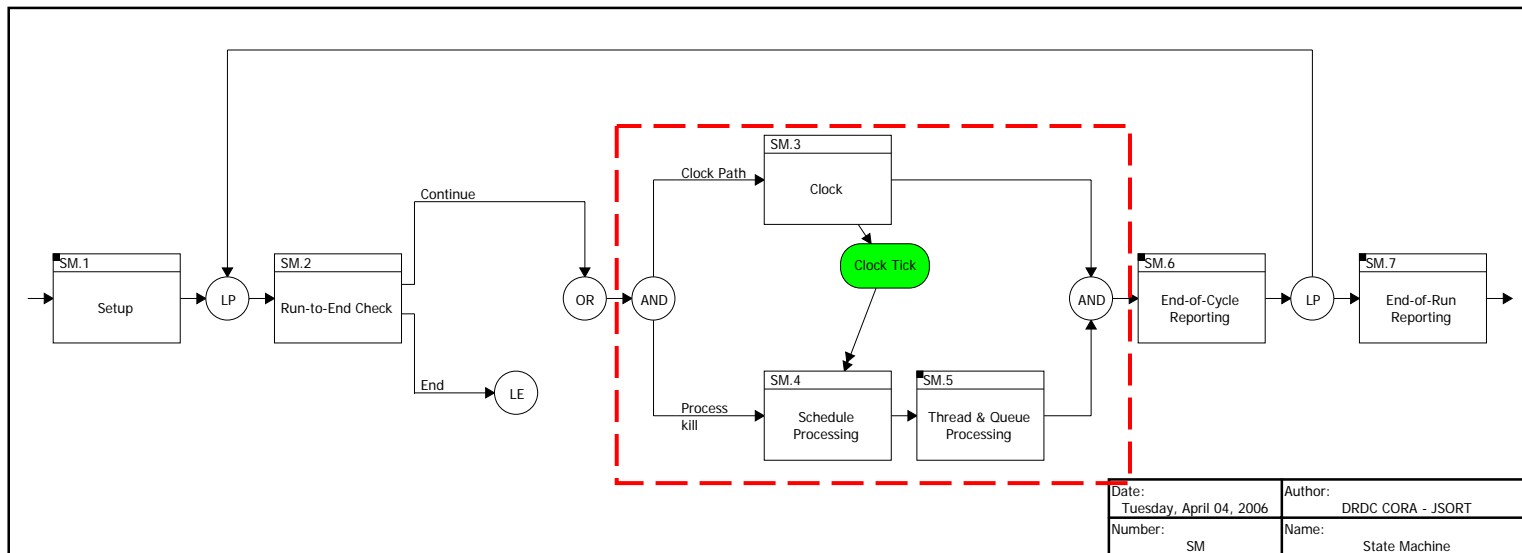


Figure 2: OPCEN SM Top Level

1.4 SMOFN Contract Summary

DRDC CORA supported the development of the SMOFN engine through a contract for specialized system engineering support. This allowed JSORT to exploit all the capabilities of CORE as well as gain access to Model-Based Capability Engineering (MBCE) expertise. The project was originally split into an initial exploratory phase and four (4) optional phases. Each phase and version of the model yielded new insights and that experience was used to scope out the requirements and development path of the subsequent phase.

Each phase was composed of a development package and a workshop (except phase 2 which had two workshops). The workshops provided a forum to get operator and researcher feedback about the validity of the model proposed at that stage and to develop the requirements for the subsequent phase. The deliverables from the contractor were based on periodic delivery of updated source code and a revised user manual at the end of each phase. The latest version of the user manual can be found at Annex A. A summary of the dates each phase was active and what was accomplished follows:

1. **Phase 1** – (January 2006 to March 2006). The first phase was an exploratory effort which used nominal data to put together a simple generic version of the SMOFN engine. The conclusion was that the SMOFN was a feasible concept and the requirements capture process began for a full-scale version of the SMOFN;
2. **Phase 2** – (May 2006 to September 2006). This was the major development phase to build the generic data file and thread structure needed to support a fully flexible SMOFN engine. This phase also included the articulation of the basic underlying conceptual framework that allows the model to be instantiated as a completely scale free design.
3. **Phase 3** – (October 2006 to January 2007). This phase involved continued testing, validation and extension of the SMOFN engine to handle Consume, Discover, External Sources and Repository;
4. **Phase 4** – (February 2006 to March 2007). Implementation of the Discover and Repository nodes was completed. A process of rapid Standard Operating Procedure (SOP) modelling was also developed and tested to improve the usability of the SMOFN, as modelled job threads could be quickly documented and validated; and
5. **Phase 5** – Not activated. The workshop assigned to this phase was used to augment phase 2.

The contract was successfully completed within the total costs and schedule laid out in the initial statement of work. The results of the contract (and the basis of this report) have been presented at two international conferences [9], [10].

1.5 Aim

This report is intended to document the functionality of the SMOFN simulation. If the investment of time is to be made to collect the data necessary to turn the SMOFN into a full simulation of the CF command structure, it is important to generate an understanding of how SMOFN works and

therefore of what data would be required. With that in mind, a secondary aim is to document the proposed data collection method, drawing from one example of an undocumented SOP, and one example of an extensively documented SOP.

1.6 Scope of the Modelling Effort

The focus of the SMOFN engine is on operator workload during product creation. Nodes other than the Producer are only modelled insofar as necessary to control data flow. Also, although the model accounts for how much time is spent on each step of a product creation job, it is not concerned with the details about how each step is performed. While the concept recognizes that Consumer OPCENs use received products to inform their decisions as to what actions are to be taken, this C2 activity is not directly accounted for in the SMOFN.

While the architecture views presented in this report were created when DoDAF was the standard framework used by the CF, they are all compatible with the equivalent views in the Canadian Department of National Defence (DND)/CF Architecture Framework (DNDAF) [11].

1.7 Layout of Report

This introductory chapter is intended to present SMOFN's predecessor, the OPCEN SM, and to describe the way forward from the OPCEN SM to SMOFN. The rest of the report is split into 2 parts. Part 1 describes the development effort to create the SMOFN and Part 2 examines the way forward to populate and run the SMOFN as a simulation of the CF Command Structure.

Part 1 begins in Chapter 2 with a description of the conceptual framework that the SMOFN is intended to simulate, including a walkthrough of the modelled logic as it applies to each node. Chapter 3 then describes the SMOFN engine itself, as it is implemented in CORE. Chapter 4 describes the data files that the SMOFN refers to in order to define the jobs and OPCENs that are to be simulated. This read-on-execution input is what allows the SMOFN to be flexible and scale-free.

Part 2 then begins with Chapter 5, which explains how data can be collected so the data files described in Chapter 4 can be tailored to reflect reality. Chapter 6 outlines two examples of the data collection process that were used as trial cases. Finally, Chapter 7 sums up with the conclusions and recommendations stemming from this paper.

Part 1 – Developing the SMOFN Engine

2 SMOFN Engine Modelling Concepts

2.1 Extension Beyond the OPCEN SM

The OPCEN SM has been extended to account for Net-Enabled interactions between several OPCENs. In essence, the OPCEN SM accounted for the work done by a single OPCEN to produce several products based on data analysis. In the SMOFN, not only are such production jobs tracked within several OPCENs, but the products they create are uploaded to a common, networked Repository so that all products can be accessed and used by any OPCEN. The SMOFN engine is an attempt to capture the essential logic that governs the way work is conducted in a general sense, but with particular emphasis on ensuring that it can be fully applied to integrated activity between networked OPCENs.

Because the OPCEN SM concentrated on job production by an OPCEN, there was not much that needed to be added there. Instead, most of the work concentrated on adding the effect of nodes outside of the Producer, as well as creating the interactions between all of the nodes. Within the Producer node, the only modification has been to allow users to create customized job threads. The OPCEN SM had assumed that every job involved a maximum of 10 steps, with fewer steps handled by assigning zero time to unneeded steps. Instead, SMOFN builds its job threads dynamically by assigning each step a unique name and linking any number of them together by referencing the names of subsequent steps until it reaches a final step called “COMPLETE”.

2.2 Conceptual OV-1

Figure 3 illustrates the concept of SMOFN as a high-level operational concept graphic, known in DoDAF and DND AF as an Operational View (OV)-1. It is a scale-free design in the sense that it is equally valid for representing interactions between nodes, or within a node, or even within a single operator’s perspectives (depending on their assigned functions). For example, every operator in every OPCEN performs the Consumer, Producer, and Discovery activities at one time or another. The major difference between instantiations of each activity in a node is that the processes become more formal as the number of individuals involved or span of control increases.

The dashed lines show the net transfer of Products, Questions, Queries, Results, and Responses, while the solid lines are a reminder that all of the transactions actually occur through the Repository, the interface to which is implemented by some form of Portal. This conceptual OV-1 diagram was used as the basis for describing the logic that controls the interactions between the various nodes in the SMOFN. More specifically, it was used to define the interactions between each node and the Repository.

This OV-1 drew inspiration from the information collection and processing sequence as illustrated in Figure 25 of [12] and uses feedback received from Maritime Forces Atlantic (MARLANT) Operations Research (N02OR), who have developed it further to suit the specific purposes of Marine Security Operations Centre (MSOC) East [13].

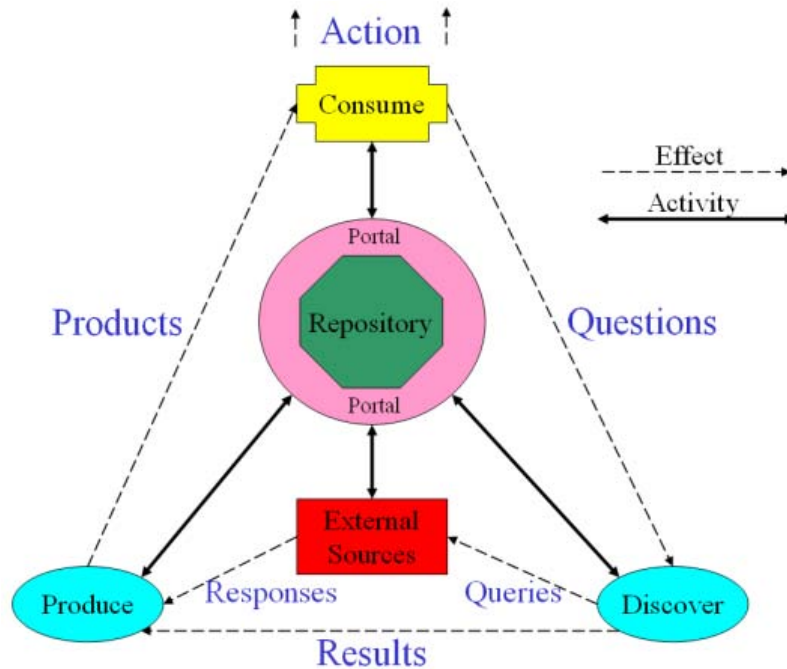


Figure 3: Scale-free SMOFN Operational Concept Graphic OV-1

Each of the nodes illustrated in Figure 3 has a unique responsibility:

1. The **Produce** activity generates products from available data inputs. The data is drawn from the **Repository** and may have been put there either as the results of the **Discover** process or as **External Sources**' responses to queries. The products are then sent back to the **Repository** for the **Consumer**;
2. The **Consume** activity receives products which it uses to maintain situational awareness (SA) and as a basis for commanding action to be taken. If the **Consumer** has questions based on the received products, it sends them to the **Discover** activity;
3. The **Discover** activity receives questions from the **Consumer** and searches to see if they can be answered using available information. Any discovered information is passed to the **Producer** to incorporate into existing or new products. When any or all of the information cannot be found, the question is converted into a formal query to request new or modified inputs from **External Sources**;
4. The **External Sources** activities receive queries and send responses to the **Producer** in the form of new data; and
5. The **Repository** stores information and acts as a medium for the other nodes to communicate. The **Repository** node contains all the information created or used by the other nodes and uses its operational rules to handle the connectivity between them.

2.3 Examples of the Process

Table 1: Examples of the Process

Node	Generic Example	MSOC Example
Producer	Assume that Operator #1 (Op #1) takes on the role of a producer who follows a series of steps when preparing a report. The report is sent to the Repository as a product to be consumed by Operator #2 (Op #2).	MSOC data analysis produces a report that includes the location of a vessel of interest (VOI) in a large area of the ocean. The producer updates the position intelligence on likely intent and time frame in the repository.
Consumer	Op #2 uses the report as a basis for their actions. If the report does not contain sufficient information or is incorrect from Op #2's perspective, a request can be made for more information. This request can be sent back to Op #1, handled internally by Op #2, or sent to a third party.	As the consumer, Comd MSOC views that information via the portal to gain situational awareness about the VOI, and notices that the data is three days old.
Discoverer	Any operator that looks for more information now takes on the role of discoverer. If the information is found, it is forwarded to a producer (who may or may not be the original producer or consumer) who then creates the new product with the found information for Op #2.	Discovery is initiated to see what newer data is available, including assets that are currently monitoring the VOI. None are assigned so a query is made to update the VOI position using external sources.
External Sources	If the discovery process fails to yield the correct information, then a formal request for help is sent to External Sources. The query includes direction to provide a single response or start automatic updates of the data. As with information found by Discovery, new data is forwarded to a Producer.	If no other information is available, then an aircraft is assigned to search for the VOI. The aircraft may, or may not, find it but either way the results of the search mission are posted to the Repository.
Producer	The Producer incorporates the external response and/or Discovery results into either a new or existing product for the consumer (Op #2). If the query involves automatic updates of the data then the producer will factor that data, where relevant, into any subsequent products.	As the Producer, MSOC data analysis uses those results in future analysis products, always with a view toward increasing the Consumer's (Comd MSOC) SA.

Table 1 describes how the roles illustrated in Figure 3 work together. The first example describes the logic as the SMOFN is designed in a generic sense. The second example is based on the

specific case of an MSOC operation, the details of which could be captured in the SMOFN data files. This is meant to illustrate how the SMOFN's generic coding can be used to study real-world scenarios.

Note that in the generic example, the only relevant factor is the switching of roles between producer, consumer and discoverer so the SMOFN interpretation is equally valid for actions by a single person or separated between multiple positions.

2.4 Logic Within Each Node

So far, this chapter has focused on describing what each node is responsible for, and how it interacts with the nodes around it. This section will provide the details of the logic that must be executed by each node in order to accomplish its tasks.

2.4.1 Producer Logic

The focus of the OPCEN SM was the logic controlling Producer threads. Because the SMOFN includes multiple OPCENs, it expands on this functionality by including a check for the arrival of new jobs at each one. Each OPCEN then uses the OPCEN SM logic rules to assign operators to jobs. First, each job is examined in order of priority and assigned to available operators with the skills necessary to perform the job. Some jobs that are in progress from the previous time step can be interrupted if a higher priority job requires the same operator, or a job step considered to be generic can be interrupted to free a skilled operator so they can move to a job requiring their skill, leaving the generic job for someone else.

Products are created with the objective of enabling *sensemaking*³ by the Consumer, so it is the Consumer who determines the utility of any product. However Producers estimate the utility of jobs as they are worked upon so those jobs can be prioritized. Jobs are abandoned when their utility decays faster than the operators can increase it. Jobs are also abandoned if they cannot be completed by their drop-dead time, which is specified separately for each job. If a job's drop-dead time is omitted in the input files, it will instead be calculated based on the job's arrival time at the queue and a standard allowable production time for that job type.

2.4.2 Consumer Logic

The Consumer performs the direct C2 activity by receiving information products from the Producer, ordering actions to be taken, and issuing questions when there are gaps in the Consumer's SA. In terms of SMOFN execution, the Consumer node's job is very straightforward. Each time a product is received by the Consumer, there is a chance, based on the type of product received, that a question will be generated. The Consumer reviews each product received and then possibly, based on the set probability for the product type, generates questions after a certain delay time. These questions are sent to the Repository to be passed on to the Discover node.

³ Sensemaking is defined as "the process by which individuals (or organizations) create an understanding so that they can act in a principled and informed manner" [12].

2.4.3 Discover logic

The Discover node is responsible for answering questions generated by the Consumer. This can actually be done by any operator, including the original Consumer or Producer. In real life, this tends to be done through a search for existing data. In modelling terms, it is handled through job threads similar to those used by the Producer. In fact, most of the logical scripting used by the Producer threads is shared with the Discover threads except for a few important differences. First, the Discover logic does not track utility because that will be determined by the analysis job performed by the Producer on the newly discovered data. Also, the Discover thread either returns new data that is sent to the Producer to help answer the Consumer's question, or tasks External Sources to collect that data. It is also possible for Discovery to generate both outcomes, meaning that even if data is found and sent to the Producer, it is possible that more data is needed, in which case a query is also sent to External Sources.

The actual process can be to send the questions to others to resolve or try to answer the information directly. Either way, if information is discovered, the results modify or add to the existing products and affect the SA in the Consumer's mind. The Discovery process can be hard to quantify because Consumers can generate internal and informal questions that get answered without the remainder of the architecture being aware of them. Only if the initial search does not yield the needed answers will the question be converted into a formal query.

2.4.4 External Sources Logic

The External Sources node receives queries from the Discover node and is tasked with finding the information that will be sent to the Producer so that the Consumer's original question can be answered. The External Sources may:

1. Already have the information and only need to find and deliver it;
2. Collect new data; or
3. Forward the query on to other External Sources.

Exactly how the requested information is found is not within the scope of the SMOFN. What is important is how much information is found (in terms of file size) and how long it takes to find it. The found information will typically be in the form of raw data, which is then sent to the Producer to trigger a new analysis job.

2.4.5 Repository Logic

The repository is the medium between all of the other nodes. Its job is to transfer products from Producers to Consumers, questions from Consumers to Discoverers, queries from Discoverers to External Sources, results from Discoverers to Producers, and responses from External Sources to Producers. The SMOFN handles all of these transfers in a similar manner, with nuances to account for differences between types of data or nodes. More particularly, it also handles meta-information (information about the information transferred) relevant for the receiving node to execute its logic appropriately.

2.5 Producer-Repository-Consumer Model

The above description assumes the existence of the Producer-Repository-Consumer model [7] to account for ways in which Producers transfer information through the repository to Consumers. The Repository itself can be composed of any conglomeration of servers and databases but they are viewed through a Graphical User Interface (GUI) that acts as the portal allowing the user to treat all the information as residing in a single Virtual Knowledge Base (VKB).

This approach also assumes that all connectivity takes place through the Repository using the portal GUI that operators can customize to address their specific needs. The Repository operational rules direct the data from sender to recipient. Every node can select the operational rules to automatically push or pull the required information between nodes, or to allow the manual push or pull by the appropriate operators. The ability to set and test the interaction of detailed data transfer rules within the simulation provides a clear indication of how these seemingly benign assumptions can influence the network's operational efficiency.

Figure 4 illustrates the ways in which Producer and Consumer nodes can communicate with each other and interact with the Repository. The most direct way for content to be transferred is for the Producer to push products directly to the Consumer. Note that during the actual execution the transfer will most often occur over a network with the Repository acting as the medium to accomplish this. Alternately, the Producer may post content to the Repository where it waits to be pulled by interested Consumers. The logic options include allowing the Producer to notify the Consumer that content is available, or allowing the Repository to use its operational rules to decide when to notify the Consumer, or leaving it for the Consumer to find when searching for new information.

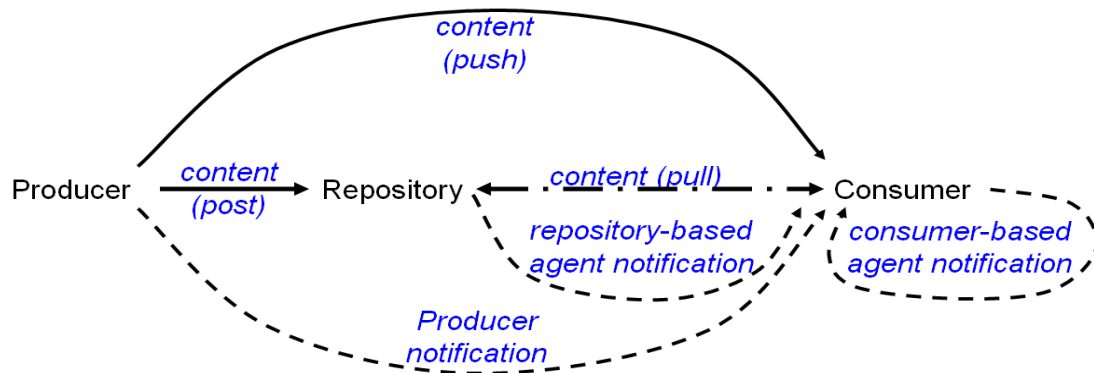


Figure 4: Producer-Repository-Consumer Paths

3 SMOFN Engine in CORE

While the previous chapter explained the process that the SMOFN is simulating, this chapter focuses on the logic controlling the SMOFN. In other words, we are moving from the conceptual (what is being simulated) to the technical (how it is being simulated).

3.1 SMOFN Top Level Description

Figure 5 is the top-level diagram that controls SMOFN execution. It is intended here to illustrate the process as a whole however larger scale views of each section of the diagram are available in Annex B as Figure 45 through Figure 48. Annex B also includes illustrations of lower levels of decomposition which are only incidental to the discussion here. Figure 5 is colour-coded so each activity can be easily cross-referenced with the appropriate node of the SMOFN OV-1 diagram in Figure 3. The addition of the new nodes is the main difference from the OPCEN SM in Figure 2. Figure 5 is segmented into five phases of activity:

1. **Simulation Control** handles the process of incrementing the clock by one time step and checking for the end of the simulation. It also includes the input of scenario data at the beginning of the simulation, before entering the time step loop;
2. **OPCEN Input** simulates the delivery of information from the Repository to OPCENs (Producers, Consumers, and Discoverers) and to External Sources at the beginning of each time step;
3. **OPCEN Activity** refers to the activity within those nodes, that is the generation of questions by Consumers, the processing of jobs by Producers and Discoverers, and the production of new data by External Sources during each time step;
4. **OPCEN Output** simulates the uploading of information from OPCENs and External Sources to the Repository; and
5. **Simulation Output** stores pertinent information in output variables after each time step and saves those variables as output files at the end of the simulation.

Unlike most activity diagrams created in CORE, time in the SMOFN does not increase as CORE steps through the process from left to right. In this case, time is actually stopped and the displayed left-to-right sequence represents the decision making process that takes place at each time step. Once the decision logic is completed SMOFN increments time another step and repeats the process.

Another key difference between Figure 5 and typical CORE behaviour diagrams is that the process logic is not completely defined simply by the diagram itself. As mentioned earlier, the non-serial behaviour of TPPU logic is not scalable when modelled by classical flowchart diagrams, so SMOFN uses scripting embedded within each activity block in the diagram to track the state of each job.

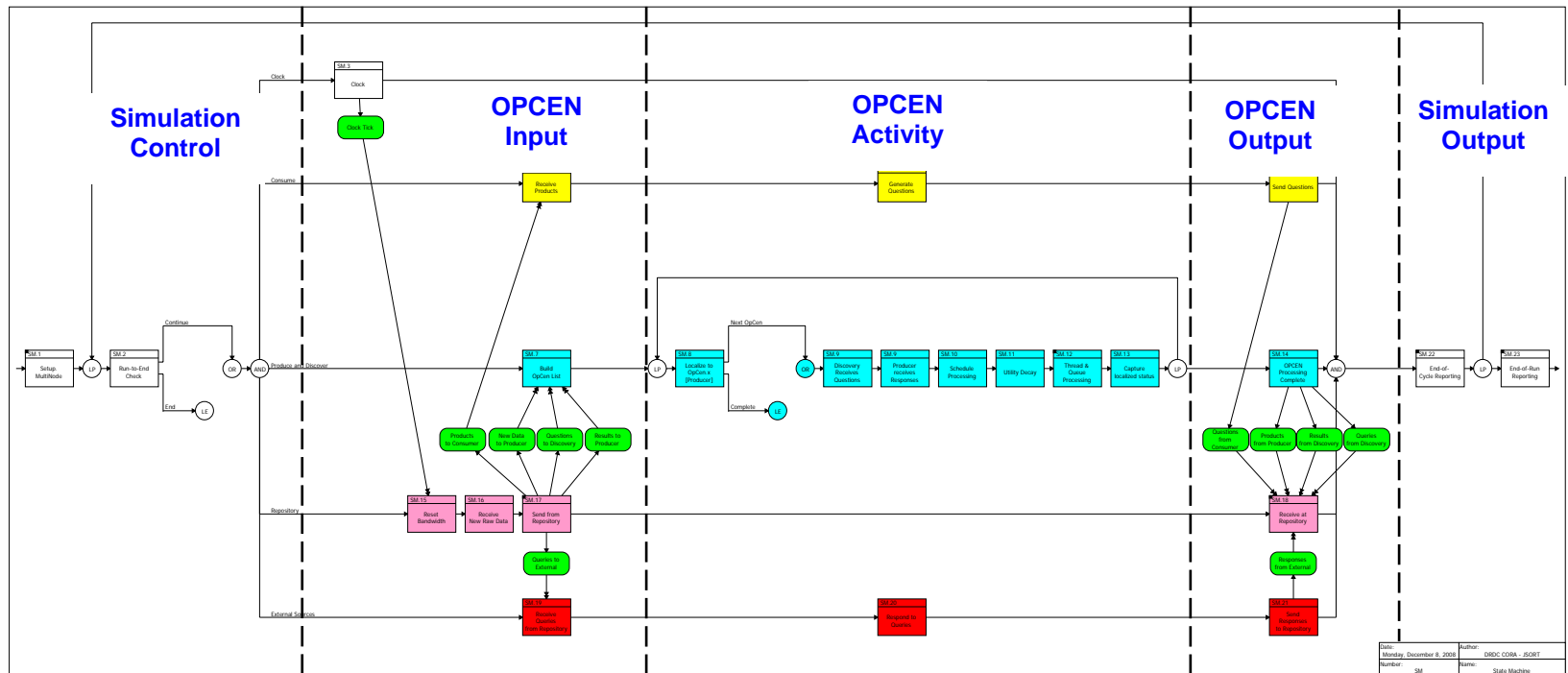


Figure 5: SMOFN Top-level Model

3.2 Logic Within Branches

The simulation control and simulation output phases control the main branch of the simulation. The other three phases are divided into branches controlling the clock and the four operational nodes. The following discussion lists the logic branches in the order shown in Figure 5.

3.2.1 Main Branch

The SMOFN logic elements shown as white boxes in Figure 5 are the simulation control and simulation output phases. When the simulation starts, the setup activity runs, reading data from input files and initializing global variables used to track data throughout the simulation. The SMOFN then enters a loop which repeats for each time step. First the Run-to-End Check executes to see if the end of the simulation has been reached (this will occur at a pre-set time or when there are no jobs left). If the end of the simulation is reached then the loop exits and the SMOFN skips ahead to the End-of-Run Reporting which outputs the data that was tracked throughout the simulation. If the end is not reached, the clock increments one time step and each node's logic is executed. Afterward, the current state of every task and every operator is saved to the global variables that form the basis of the simulation output.

3.2.2 Clock Branch

The clock activity initiates all SMOFN activity by incrementing the simulation clock and sending the trigger that starts the repository branch activity during each time step.

3.2.3 Consume Branch

The yellow boxes in Figure 5 represent the Consume activity of the SMOFN OV-1. Each time step, the Consumer node begins by receiving products sent by the Repository. The "Generate Questions" activity then goes through all the products that have just been received and may generate questions based on those products. If any questions are generated, the delay before they are actually sent is determined and the time to send them is added to a schedule. This delay is meant to account for any time between when the product is received and the question is sent out and can be attributed to several factors, including: time to study the product, time to formulate a question, and any time where the Consumer is engaged in other activity. As in real life, if the probability of generating questions is high enough, the cumulative demand placed on other nodes steadily increases during the simulation, eventually exceeding their capacity to respond.

3.2.4 Produce and Discover Branch

The Produce and Discover activities share much of the same logic so their execution is controlled together by the blue boxes in Figure 5. First a list of OPCENs is built and a loop is entered that is repeated for each OPCEN. The loop begins by choosing a new OPCEN and the "Schedule Processing" activity then looks at the job arrival schedule for that OPCEN and adds to the queue any job that arrives during the current time step. "Utility Decay" then goes through all the jobs and checks to see how old their data is. If the age of the data is such that the utility should

decrease this time step (as explained later in Section 4.2.5), that is handled here. The “Thread & Queue Processing” element goes through all jobs and assigns available operators to jobs in order of priority. It is here that the utility of jobs can increase as work progresses, or jobs can be abandoned if they cannot be completed on time (under the TPPU paradigm, jobs that were partially completed when abandoned can still be delivered for use by consumers). Produce and Discover jobs may draw from a single pool of operators so when the SMOFN sorts jobs in order of priority, it ignores whether the job is related to Production or Discovery. After these activities have executed, the “Capture Localized Status” element stores in memory any data that must be tracked into the next time step. This data includes the current state (ie. job assignment) of each operator and the current state of each job (ie. completed, abandoned, or some amount of time remaining in a given job step). After the loop has executed for each OPCEN, the “OPCEN Processing Complete” element signals to the Repository that it can now execute its logic for receiving products and queries.

3.2.5 Repository Branch

The pink boxes represent the work done by the Repository during each time step. It is this branch that receives the trigger from the clock branch telling it to begin its activity. The Repository logic then starts by resetting the global variable tracking the bandwidth between the Repository and each OPCEN, which will be decremented whenever bandwidth is used to send or receive data. Bandwidth is tracked in terms of how much data can be transferred during each time step.

Next, the Repository checks for the arrival of new raw data. This check is based only on parameters read into the model during setup; raw data received from External Sources in response to queries is handled separately. “Send From Repository” contains the logic used to send products to the Consumer, jobs to the Producer, questions to the Discoverer, and queries to External Sources. This activity takes place before each of the other nodes executes during the time step (it creates the trigger allowing the other nodes to start) so they can incorporate data as they receive it. Similarly, “Receive At Repository” waits until each node has completed its execution for the time step so it can receive data as soon as it is ready. This activity receives questions from Consumers, products from Producers, queries from Discoverers, and responses from External Sources. It should be noted that the Receive and Send activities of the Consumer and External Sources have no embedded logic scripts. Their logic is actually handled within the Send and Receive activities, respectively, of the Repository. They are shown to illustrate the Consumer and External Source perspectives of what is taking place.

3.2.6 External Sources Branch

Finally, External Sources from the SMOFN OV-1 are simulated with the red boxes. They receive queries from the Repository, respond to those queries, and send the responses back to the Repository. As with the Consumer, the process required to respond to queries is not tracked in detail, only the amount of time required and the type of raw data returned (specifically, the type of analysis job that will be triggered at the Producer when it receives that data) are currently handled within the simulation.

4 SMOFN Input Data Files

One advantage of the SMOFN is that the model itself is only concerned with the execution of the operational logic. All operating parameters, such as the number of operators at an OPCEN and their skill sets, or the number of jobs to be done, are read from data files during the setup stage of the simulation. This allows a single version of the model to be used across a limitless variety of scenarios.

The explanations that follow broadly describe what is contained in each input file but their detailed instantiations are left to the User's Manual in Annex A.

4.1 Simulation Setup

The first data file read into the SMOFN points to the data path where the scenario files reside. The simulation will prompt the user to identify the location and name of this file. It will then identify the path to all other files that need to be read. The scenario files include those needed to setup the simulation and those used to describe the OPCENs that are simulated. The setup files include:

1. The time at which the simulation will stop;
2. The day of the week and time that is represented by the beginning of the simulation;
3. A switchboard to allow the analyst to select values for a list of generic operational rule settings; and
4. A list of OPCENs and the data paths containing their setup information.

4.1.1 Switchboard

There are six rules that can be specified in the switchboard, three that are specified by answering yes or no to a specific question, and three that require numerical answers.

The first, and simplest, yes or no question is whether there is any chance of a job resulting in Nothing Significant to Report (NSTR). If yes, the probability of this happening is specified later as it is specific to each thread type.

The next question covers the situation where a job would normally be abandoned because it cannot be completed on time, but there is still time to complete at least one more step. The question is whether the step should be completed so that the product is as complete as possible when time runs out, or whether the job should be abandoned immediately.

The final yes or no question asks if a job should be abandoned when its utility appears to be decaying due to age faster than it is accruing due to work performed by the operator.

The first numerical question deals with the fact that if two jobs have the same priority, one of them may be treated as a higher priority if it is considered rushed. The number specified here defines the ratio, expressed as a percentage, of remaining slack time (the amount of time that a job can be ignored and still be completed on time) to remaining job time at which a job is considered rushed. For example a job may be considered rushed if its slack time is equivalent to less than 50% of the time it will take to be completed. If two jobs of equal priority are both considered rushed, the one with the lowest ratio of slack time to remaining job time will be treated as the higher priority.

The last two questions deal with a job step that is being restarted after having been abandoned between posts. It is assumed that the job step may be restarted from the last post rather than from the beginning, but that some time must be taken by a new operator to become familiarized with the work already done. One question asks how much time must be spent by a new operator to become familiar with work already done before they can begin progressing on the actual job. This is expressed as a percentage of the amount of time between job posts. The other question examines the situation where an operator comes back to his own work and asks how much time (in minutes) could that operator have been away before needing to re-familiarize himself with the job and suffering the same time penalty as a new operator would.

4.2 OPCEN Inputs

The parameters for each OPCEN are defined by five different files that are read into the SMOFN. The following files are found in the OPCEN data paths that were read from the first simulation setup file described in the previous section:

1. A summary of thread types;
2. The step-by-step definitions of each job thread;
3. An events list;
4. A matrix of operator skills; and
5. A set of utility decay function definitions.

4.2.1 Thread Types

The thread types file defines the overall characteristics of each job type that is processed at the particular OPCEN. The details specified here reflect the job as a whole, whereas the characteristics of each step within the job thread are specified in the thread definitions file. Each entry is identified by thread name and priority. It is important to note that in the case of specific jobs, any value for priority is acceptable however when filling in data for thread types, it is not feasible to include every possible value of priority. Therefore, if an instance of a job is given a priority that is not included in the appropriate thread type definition, it will be treated as the nearest priority value for which the thread type has been defined. The data used to describe a thread includes the following fields:

1. Percentage of jobs that will result in NSTR;
2. Default slack time for the job, representing the amount of time a job could sit idle without becoming invalid;
3. Number of steps an operator can look ahead to estimate the expected utility gained over time spent on the job;
4. Whether the job is redundant with other jobs of its type (jobs that are redundant will be ignored if another job of the same type is in progress when a new one arrives in the queue – data-driven jobs tend to be specific to the newly arrived data, whereas SOP-driven reports tend to be redundant);
5. Utility of a job that returns NSTR. Instinctively this may be zero, but there are certain jobs where it is worth something simply to know that there is NSTR; and
6. The details of aggregating other jobs into the current one. Used by the SMOFN to properly account for the aggregation of several completed products into one new job. For example, several imagery analysis jobs being reported together in one Situation Report (SITREP). The following details need to be specified for the aggregating job thread:
 - a. Name of the job step where aggregation will take place;
 - b. Types of completed jobs to aggregate, and for each job type:
 - i. The amount of time that must be added to the aggregation step for each job aggregated; and
 - ii. The percentage of the aggregated job's utility that carries over to the aggregating job.

4.2.2 Thread Definitions

The last file describes how the job threads are constructed, in terms of how the SMOFN will handle each step within each job. This file uses one line of data for each step of each thread type and these types must be the same, by name and priority, as those listed in the thread types file. Each line of data has 12 entries, identifying:

1. The thread type (including name and priority);
2. The name of the current step, which must be unique within the thread type;
3. The time required to complete the step;
4. The skill required;
5. The number of times the product's utility is updated by being saved to the Repository during the step (these are the "Posts" in TPPU);

6. Whether or not the step is allowed to be interrupted;
7. Three fields representing the minimum, mode, and maximum of a triangular distribution used to calculate the utility that is achieved by the end of the step;
8. The file size of the resulting product (used to track uploading to the Repository and subsequent distribution to Consumers);
9. The type of step. This is specified in terms of which logical script to execute within the SMOFN and therefore must correspond to a type that the SMOFN has been scripted to understand (these acceptable types are described below); and
10. The name of the next step to move to after the current one.

The “Type of Step” entry warrants further discussion. Seven types of steps are currently understood by the SMOFN, and they are identified here by number. The first three of these step types previously existed in the OPCEN SM and apply to both Production and Discovery:

1. No decision is made at the end of this step, once the it is complete the job will simply move on to the next step;
2. NSTR decision: if the job is marked NSTR, it happens at the beginning of this step, after which all subsequent steps are skipped; and
3. Final step of any Production job, after which the job is marked complete and removed from the queue.

The final four step types are unique to Discovery jobs:

4. A decision as to the results of the Discovery process, which has three possible exits, each identifying which type of step should follow:
 - a. No data exists,
 - b. Desired product is found, and
 - c. Some data is found but it is insufficient.
5. In the case that no data exists, the Discovery job will create a request for new data that will be sent to External Sources;
6. If the product is found, it must be analysed and put in the context of the originating question, so a new Production job is added to the appropriate OPCEN’s schedule;
7. If insufficient data is found, then the Discovery will do a combination of the previous two steps. It starts by sending a request to External Sources for more data, and in the mean time, a new Production job is added to the OPCEN schedule to analyse whatever data already exists.

4.2.3 Events List

The purpose of the events list is to define what jobs will be added to the current OPCEN's job queue, and when. The main entries for each job are the thread name, priority, and the time at which the job will be added to the queue. In the OPCEN SM, this list would include all jobs that the OPCEN would be expected to process over the course of the simulation. In the case of the SMOFN it should only include jobs that are based on OPCEN operating procedures. Other jobs, based on the arrival of new data are more appropriately listed in the schedule of new jobs that the Repository will send to each OPCEN (described in the next section).

The utility of the job will typically decay based on the age of the raw data the job is created from. To account for this, the file includes a field to specify when the data was collected so that the utility decay can be calculated from that point in time. Also, each job has a standard slack time based on thread type, but the events list includes a field to set a specific deadline time which overrules the standard slack time.

4.2.4 Operator Skills

The SMOFN refers to the operator skills matrix to assign operators in an OPCEN to queued jobs. Once assigned, this matrix also identifies how effectively operators do their work. The operator skills matrix identifies each operator in an OPCEN by a "name" that is a string containing only letters and numbers. The information tracked for each operator includes:

1. Their speed and quality of work (both expressed as percentages where 100% speed means jobs are done in the standard required time, and 100% quality means the operator adds the expected utility to jobs they work on);
2. The order in which they are assigned to generic work;
3. The percent chance that they are available to take on generic work when they are not already assigned any specific job. For example, supervisors may not be performing any SMOFN-tracked task but may be too busy maintaining SA to pick up a generic task; and
4. A list of their skills, beginning with their primary skill and allowing up to ten entries. When a queued job requires a particular skill, the SMOFN will look to each operator's first skill, then each operator's second skill, and so on until an available operator with the required skill is found.

4.2.5 Utility Decay Curves

The utility decay curves are used by the SMOFN to describe how the utility of products decreases as their data ages. Each line begins with a thread name and priority (the combination of which must match those in the thread types file) and a time expressed in units of the simulation's time steps. Each line then ends with three numbers representing the lower bound, mode, and upper bound of a triangular distribution. The logic of the utility decay is that after the product of a particular thread type has aged by the specified amount of time (age is counted from the moment the raw data was taken), its potential utility can be calculated by the specified triangular

distribution. The SMOFN logic is defined so that potential utility never increases as products age, even if the distribution is constructed to allow it. This works in a similar manner to the accrued utility (see [8]) except that it decreases as a function of age instead of increasing as a function of steps completed. The actual utility is calculated by multiplying the utility accrued from work done on the job by the potential utility based on product age.

4.3 Repository Inputs

A separate set of data files is used to describe the operating parameters of the Repository after the OPCEN initialization data. These files are:

1. The schedule for the arrival of new data;
2. The schedule for the arrival of new products created outside the simulation;
3. A matrix organizing the delivery of products to the appropriate Consumers;
4. A list of each OPCEN's bandwidth;
5. A list controlling the generation of questions;
6. A list controlling Discovery responses to questions; and
7. A list controlling External Sources' response to queries from Discovery.

4.3.1 Data Arrival Schedule

The data arrival schedule is similar to the events list for each OPCEN. This file is used to simulate the fact that many jobs within an OPCEN are triggered by the arrival of new data in the Repository, rather than by the OPCEN's own operating procedures. Each line specifies:

1. The time that the new data was created;
2. The time at which it will arrive at the Repository to be delivered to the Producer;
3. The source of the data (for example a reconnaissance unit);
4. The type and priority of the job thread that will be generated;
5. The deadline for the data analysis job;
6. The Producer OPCEN to which the data will be sent for analysis;
7. The delivery method (i.e. push or pull); and
8. The size of the file that must be delivered.

4.3.2 New Product Arrival Schedule

The schedule of outside products defines the arrival of products from Producers that are not tracked by the model. Product arrival is an important consideration for the SMOFN and is used to simulate how one OPCEN (or several) reacts to products created by other OPCENs, without having to model those OPCENs or their own product creation processes in detail. For example, a model of Canada Command might include SITREPs received from the MSOCs, without needing to simulate the MSOC process to create the SITREP. These products can then be forwarded to Consumer OPCENs as though they were created within the simulation. The required data includes:

1. The name of the producing OPCEN;
2. The type of job (including the priority rounded to the nearest appropriate thread type);
3. The specific priority;
4. The file size to be transferred;
5. The time at which the original data was collected;
6. The time at which the product will arrive at the Repository;
7. The status of the job producing the product (either complete or the end of a specific job step, in which case the product corresponds to the most recent update);
8. The utility achieved due to work on the producing job; and
9. The decayed utility due to the data's age.

4.3.3 Delivery Matrix

The SMOFN uses the delivery matrix to identify what products are sent from specific Producers to specific Consumers. This applies equally to products created within the SMOFN simulation as well as to products identified in the new product arrival schedule. This file is the main source of information for the SMOFN to account for data sharing between OPCENs. Each line of data contains:

1. The product type;
2. The name of the Producer and Consumer OPCENs (only one Producer and one Consumer per line; products sent from one Producer to multiple Consumers must have one line for each Consumer);
3. The percent probability that a product of the given type will be sent to the specified Consumer when created by the specified Producer; and
4. The method of delivery (push or pull; automated or manual).

4.3.4 Bandwidth

The bandwidth file is very simple, containing only a list of OPCENs that connect to the Repository along with their bandwidth. This identifies to the SMOFN the amount of data that can be transferred between each OPCEN and the Repository during a single time step. The External Sources node must also be included here as a single separate entry, in addition to the specific OPCENs that are only recognized by the Produce, Consume, and Discover activities. Units are at the discretion of the analyst, but must be consistent with the file sizes for products identified in the thread definitions and the new product arrival schedule, as well as raw data in the data arrival schedule.

4.3.5 Question Generation

The questions file draws from the combinations of product types and Consumers that come out of the delivery matrix. This information is referred to whenever a product is sent to a Consumer so that questions are generated as appropriate. For each such combination, the questions file identifies to the SMOFN:

1. The percent chance that the Consumer will generate a question;
2. The file size required to contain that question; and
3. The amount of time that the Consumer will take between receiving the product and sending out the question.

4.3.6 Discovery Requests

The discovery requests file is similar to the questions file except that is referred to whenever Discovery receives a question. Based on the type of product that inspired the question and the OPCEN that the question came from, this file identifies:

1. The probability that Discovery will provide an answer; and
2. The type of job that will be triggered at the Producer if results are available.

4.3.7 External Response

The external response is again similar to the questions and discovery requests files. In this case, whenever a query is received by external sources, this file identifies:

1. The probability that they will respond;
2. The size of the file containing the response;
3. The time required to respond; and
4. The type of job that will be triggered at the Producer by the response.

Part 2 – Populating the SMOFN Engine

5 Data Collection via Process Modelling

The initial contract concern was on building the SMOFN framework so that the underlying operational logic could execute properly. This meant that simple notional data sets were adequate to test the model logic and demonstrate the dynamic aspects of the SMOFN engine.

The next concern is the collection of the data required to populate all of the initialization files described in Chapter 4. This data must be representative of the operating parameters of any OPCENs that are to be simulated, as the goal is to accurately simulate data flow between these. Emphasis needs to be placed on describing job threads (especially the Producer steps) that involve multiple OPCENs, such as the process to report and respond to events in theatre or the preparation of high-level daily briefs. Another major component is the composition of various OPCENs, as far as the number of operators on shift and the skills they are trained in. Although much of this data will be classified, the SMOFN engine itself remains unclassified as it is saved in a separate file from any classified data which it would read upon execution.

This chapter outlines the method that the authors have used, and recommend, to collect the data required for the input files described in Chapter 4. Through this process, two job threads were created for the SMOFN, which represent processes carried out by the Strategic Joint Staff (SJS) National Defence Command Centre (NDCC). While this chapter focuses on the data collection process, the NDCC job threads that were created are presented in Chapter 6.

5.1 Description of Proposed Method

The planned process for building the future operational architecture of the CF command structure with the help of SMOFN is illustrated in Figure 6.

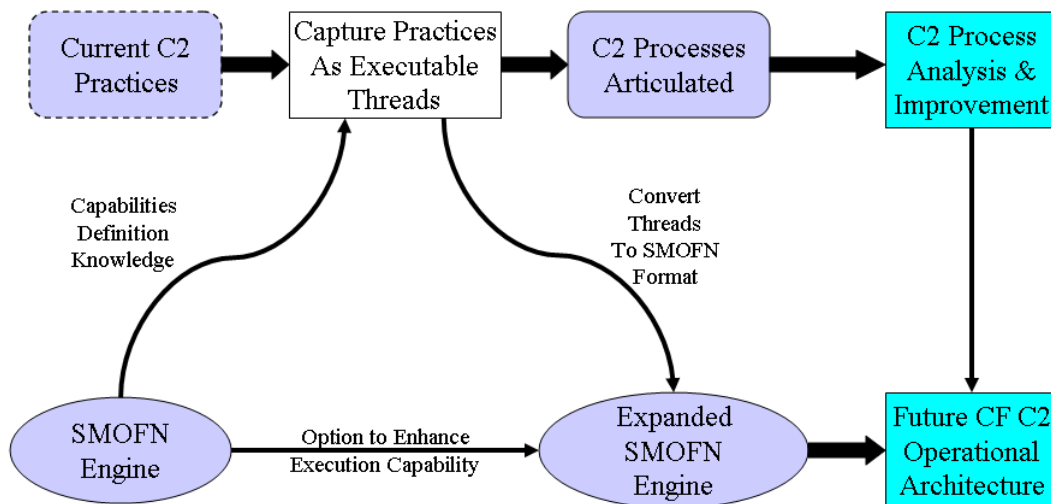


Figure 6: Planned Operational Architecture Creation Process

The work starts with the analysts converting whatever evidence exists (typically documentation or verbal descriptions from operators) of the current C2 practices into model form. In theory these should be SOPs that execute as a thread from start to finish. The actual experience is that most SOPs tend to be incomplete or inaccurate so their logic will not actually execute as described. The knowledge gained through previous work with the SMOFN engine is used to identify and address the gaps so that the modelling process eventually captures fully executable threads. These threads can then be used to create input files to expand the scope of the SMOFN. If necessary, the execution logic can also be enhanced by incorporating new operational rules as the analysts become aware of them through the modelling process. The resulting model can then be used to articulate the C2 processes as refined SOPs, which can also be used for further analysis and improvement.

The fact that SOPs tend to be incomplete or inaccurate is not just a problem for the modellers, but also for any operators who are newly posted into their position. The payoff from their perspective is that C2 practices are quickly captured, documented and converted into validated SOPs which can be passed on through the next round of postings. The review process allows the operators to increase their awareness about the C2 processes that may then expose deficiencies in current SOPs and practices. Often this is as simple as operators having developed their own methods which work better than those in the SOP, but which have never been captured on paper. The process also gives the operators an opportunity to identify ways to synchronize efforts and de-conflict duplicative work that highlights the time and resource expenses of poor C2 practises. The modelling and analysis also allows them to better assess the impact of proposed SOP changes and make an educated decision as to how to move forward. In addition, inclusion of these revised threads in the SMOFN allows them to be analysed from the perspective of concurrent activities. The cumulative effect is to improve current C2 processes as well as identify shortcomings that require new or modified technology.

The payoff from the modeller's perspective is the creation of a more complete definition of C2 processes, which leads to a better articulation of the targeted CF C2 operational architecture. As an example, the Integrated Command and Control System (IC2S) Project would use this process to access data of operational C2 processes that include previously undocumented practices and newly validated SOPs needed to populate their CF C2 Operational Architecture model. Otherwise IC2S would have to refer to existing SOPs that may be invalid or out of date.

The above process is also an important aspect of the Verification, Validation, and Accreditation (VV&A) of the SMOFN engine. If the executable threads captured from the operators are correctly transformed into inputs to the SMOFN, then the model should be able to replicate representative results that have been observed during operations. Once the VV&A process is complete for individual threads is it reasonable to start analyzing the SMOFN parameters for cumulative effect of interactions between threads.

6 Trial of SOP Modelling Process

A major discussion item that came out of the start of the Phase 4 workshop was the need to see how the SOP modelling process actually works. The consensus was that it would be easier to understand if one or two examples could actually be modelled during the workshop. The workshop participants then selected two candidates of important C2 practices; one had no organized SOP while the other SOP was considered to be the most complex one that had been documented. The thought was that if these two cases could be successfully modelled it would provide ample evidence as to the viability of the modelling approach.

In each case the approach taken was to capture information about the job thread within CORE and then read through the output transcripts with the operator Subject Matter Expert (SME) to make sure the model executed as intended. Once the threads were finalized, they could be translated into the format used to describe Producer activity in the SMOFN.

6.1 Case 1 – eBrief

6.1.1 Context Leading to Approach

The first case was an undocumented C2 practice used by the NDCC night watch to prepare the daily Electronic Briefing (eBrief) for senior commanders. The actual modelling process took two days of work by the SME and two analysts.

On the first day the lead analyst facilitated the data collection by querying the SME about the steps in the process. The data collection involved the operator SME describing the process step by step with estimates of personnel resources, timings and duration distributions. The other analyst followed the discussion while entering the data elements and proposed logic into CORE.

The second day was used to lead the SME through the modelled logic to correct misconceptions or articulate key sub-processes. The modelled eBrief logic was then tested using CORE's embedded simulator. The model ran once corrections were made for inconsistent logic and then the timeline produced by the simulator was checked with the SME to see if the model actually executed as intended. This highlighted a few cases where activities were being handled out of order and these were easily corrected. It also highlighted where process and resource bottlenecks were disrupting the preparations.

The deliverable was a DoDAF OV-6 product generated within CORE. The OV-6 is the "Operational Activity Sequence and Timing Descriptions" [2] which shows the sequence of operational activities within an architecture as well as how those activities respond to received inputs. The product included all of the diagrams in Annex C along with a detailed description of the characteristics of each element in the diagram, including values entered as attributes and relationships. This comprehensive record can be used to guide the writing of validated SOPs.

6.1.2 Description of the Modelled eBrief Process

The top-level diagram of the modelled eBrief process is illustrated in Figure 7 and is composed of five major branches of parallel activities:

1. **Timekeeping:** A clock on this branch creates timed triggers to initiate activities as necessary;
2. **Daily Preparation Process:** The main focus of activity;
3. **Wrap-up & Review:** The final activities used to get final approval of the briefing content;
4. **CDS Brief:** An activity to prepare extra slides for the weekly briefing to the Chief of the Defence Staff (CDS); and
5. **Other e-Portions:** Extra minor activities that occur during preparation

The majority of the work in Figure 7 is fairly straightforward but the branch activities all come together at the red circle when the eBrief is reviewed and approved during a short period between 0530 hrs and 0800 hrs.

The red-circled briefing approval element in Figure 7 is decomposed in Figure 8. The approval process covers 8 serially aligned events with 5 triggers indicating when each activity begins. This makes for a very intensive process with very little time to handle any unforeseen problems or make up for slippage caused by prior activities. The only saving grace with this process is that all participants realize the goal is to produce a “best effort” and minor inconsistencies in the eBrief can be corrected after the actual briefing.

The OV-6 diagrams for the other documented eBrief sub-processes are contained in Annex C.

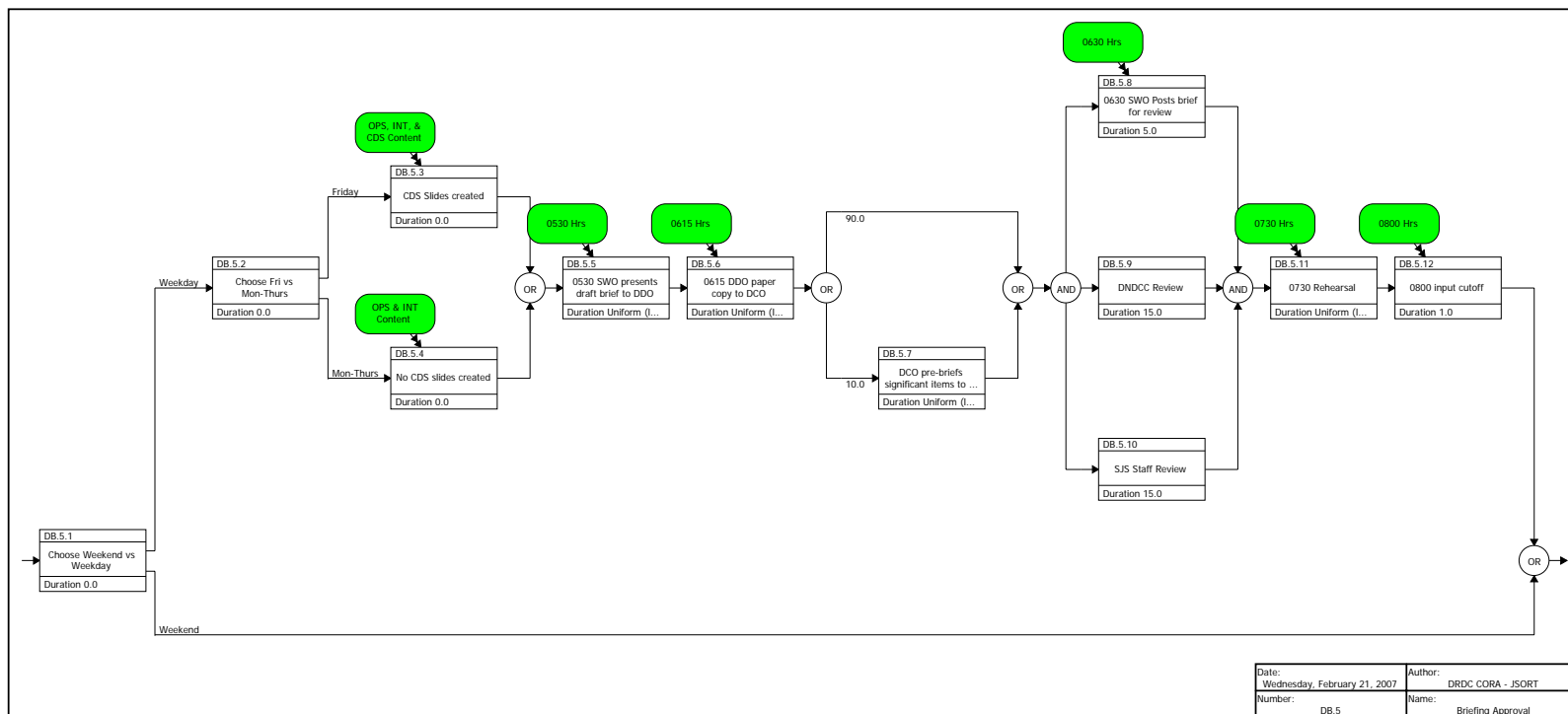


Figure 8: Final Review and Approval Process for SJS NDCC eBrief

6.2 Case 2 – Casualty Reporting

6.2.1 Context Leading to Approach

The second case describes the large number of actions currently required from several nodes when handling serious casualties during deployed operations. It is formally referred to as the “SOP Matrix for Action in the Event of Death/Serious Injury (International Ops)” but for simplicity this report refers to it as “casualty reporting”. A quick review of the documented SOP showed that it contains 180 activities, listed in 12 serials, spread over 7 operational entities as summarized in Table 2. Note that the SOP refers to a node called “NMR” but it has no indication of what this acronym stands for, and the SME was not familiar with it is not within the NDCC.

Table 2: Distribution of Unit Activities per Serial as Listed in SOP Matrix

Serial	Total	CDA	NDCC	SJS	CEFCOM	CANOSCOM	COMDS / ECS / L1	NMR (?)
Total	180	28	22	20	42	19	29	20
1	14	4	3	1	2	1	2	1
2	20	5	2	3	6	2	1	1
3	26	4	7	4	5	1	3	2
4	15	3	1	1	4	2	2	2
5	25	3	2	2	5	4	4	5
6	15	3	1	1	3	1	3	3
7	12	1	1	2	3	1	2	2
8	8	1	1	1	2	1	1	1
9	12	1	1	1	4	1	3	1
9a	3	0	0	0	0	1	2	0
10	8	1	1	1	1	1	3	0
10a	6	0	1	1	2	1	1	0
11	8	1	1	1	2	1	1	1
12	8	1	0	1	3	1	1	1

The SOP, as documented, inadvertently implies that each node follows its list of activities in a given order and that only one serial is active at any time. In other words, each node must go through the same number of Serials, and every node must finish Serial x before any node can begin Serial x+1. The SOP also provides no indication about what activities can/should be done simultaneously or cooperatively, or which activities cannot begin without first receiving information from other completed activities.

The Casualty Reporting process differs from the eBrief case in that a SOP matrix already exists that can be used to start building a model in CORE. The analyst prepared an initial version of the model based on the SOP matrix that implied a model structure as illustrated in Figure 9. The analyst then met with the operator SME to go over the logic. Together they worked through the modelled logic so it reflected the way the operators actually make the SOP work. They also entered any known attribute and relationship data associated with each element.

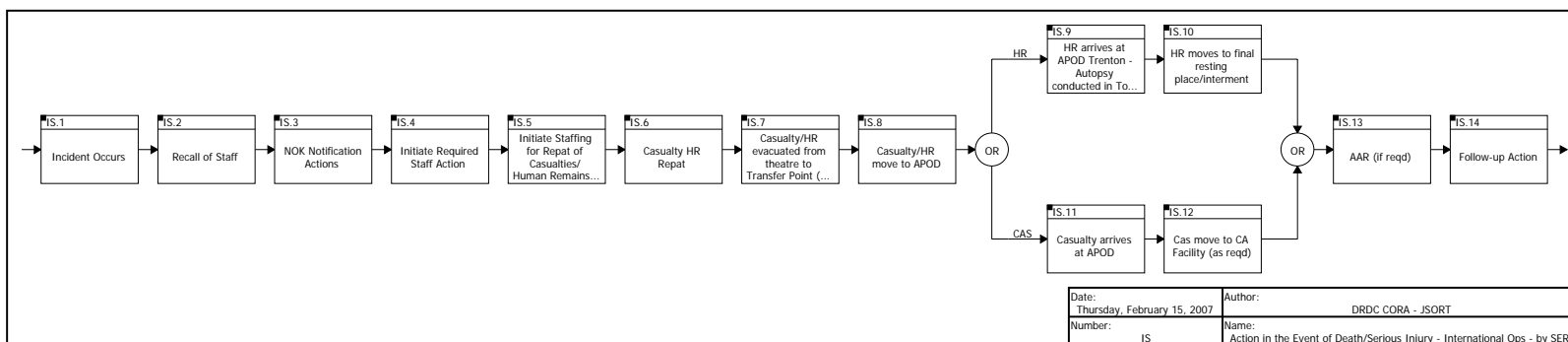


Figure 9: International Casualty Reporting Model Implied by the SOP Matrix

The modelling process required roughly 10 hours spread over five days to build the current version of the model. Table 3 shows the differences between the SOP matrix version and the one modelled with the operator SME using CORE. The process only changed the total number of activity elements by about 6% but it also added 32 information exchanges needed to coordinate activity between units (the original SOP had no information on data transferred between nodes or activities). Of those information exchanges, 30 behaved as triggers where the information must be sent before the receiving activity can proceed. The two exceptions involve information being sent from Deployed Ops and NMR to Canadian Expeditionary Forces Command (CEFCOM), and are identified by the asterisks (*) in Table 3.

Table 3: Comparison of SOP Matrix and SME Modelled Activities & Triggers

Operational Node	SOP Listed	Modeled	Triggers and Data Transfers <u>That Were Added</u>		
	Activities	Activities	Total	Out	In
Total	180	168	32	14	18
Deployed Ops (CDA)	28	30	7	6*	1
NDCC	22	20	7	1	6
SJS	20	17	3	1	2
CEFCOM	42	41	9	3	6*
CANOSCOM	19	18	2	1	1
COMDS/ECS/L1	29	26	3	1	2
NMR	20	16	1	1*	0

6.2.2 Description of the Modelled Casualty Reporting Process

Figure 10 illustrates the top level of the modelled casualty reporting process, based on the SME interviews. It draws from Figure 9 in that it identifies the Serials in order; however several of those Serials have been combined as they can be progressed simultaneously. The only real time restrictions are that Serials 1 through 8 must be completed before the casualties' arrival at the airport of disembarkation (APOD), Serial 9 must be completed between arrival at APOD and movement of the casualties, and Serials 10 through 12 must be completed after the movement. As shown in Figure 9, the SOP versions of Serials 9 and 10 contain two variations based on whether the casualties in question involve death and the transport of human remains (HR) or injury and the transport of live casualties (Cas). These variations are maintained in Figure 10, but a third branch is added to allow for the cases involving a combination of casualties of both categories.

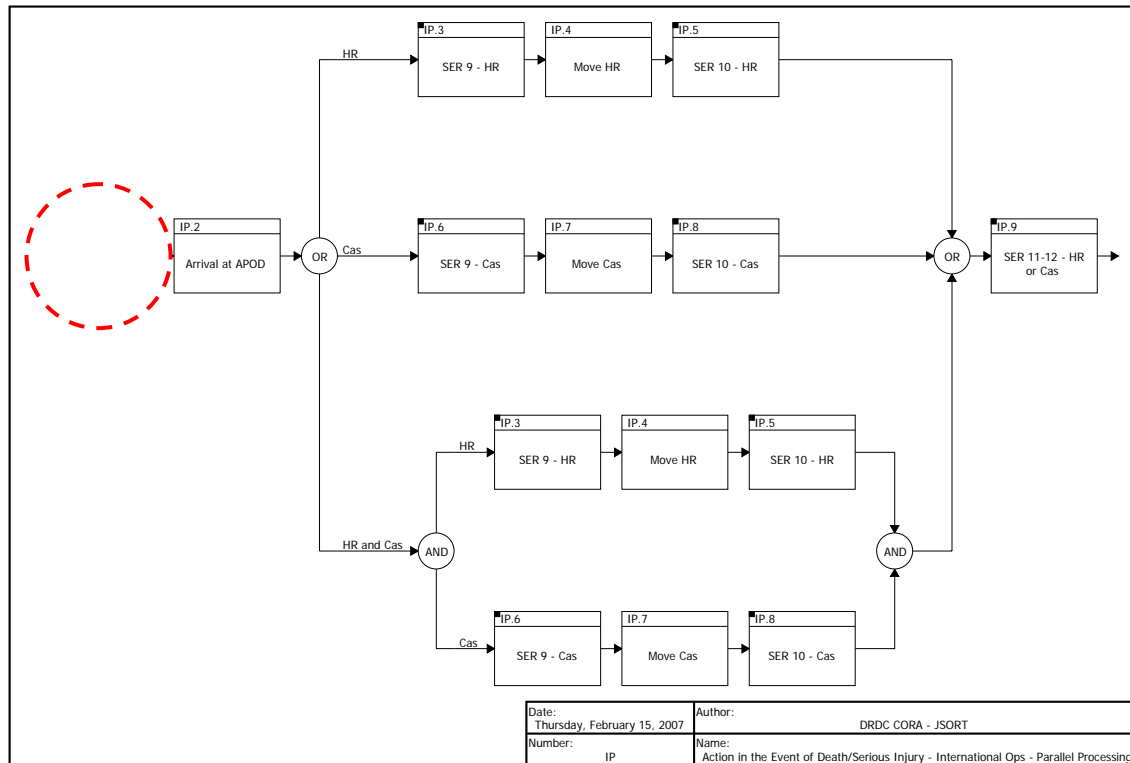


Figure 10: Top Level International Casualty Reporting Model (SME Version)

Figure 11 shows the first level decomposition of Serials 1-8, the circled activity in Figure 10. It is broken down by organization to show that each of these organizations performs their activities independently. However, several information exchanges must be completed between organizations in order for them to fulfil their responsibilities. These information triggers are represented by the green bubbles in Figure 11.

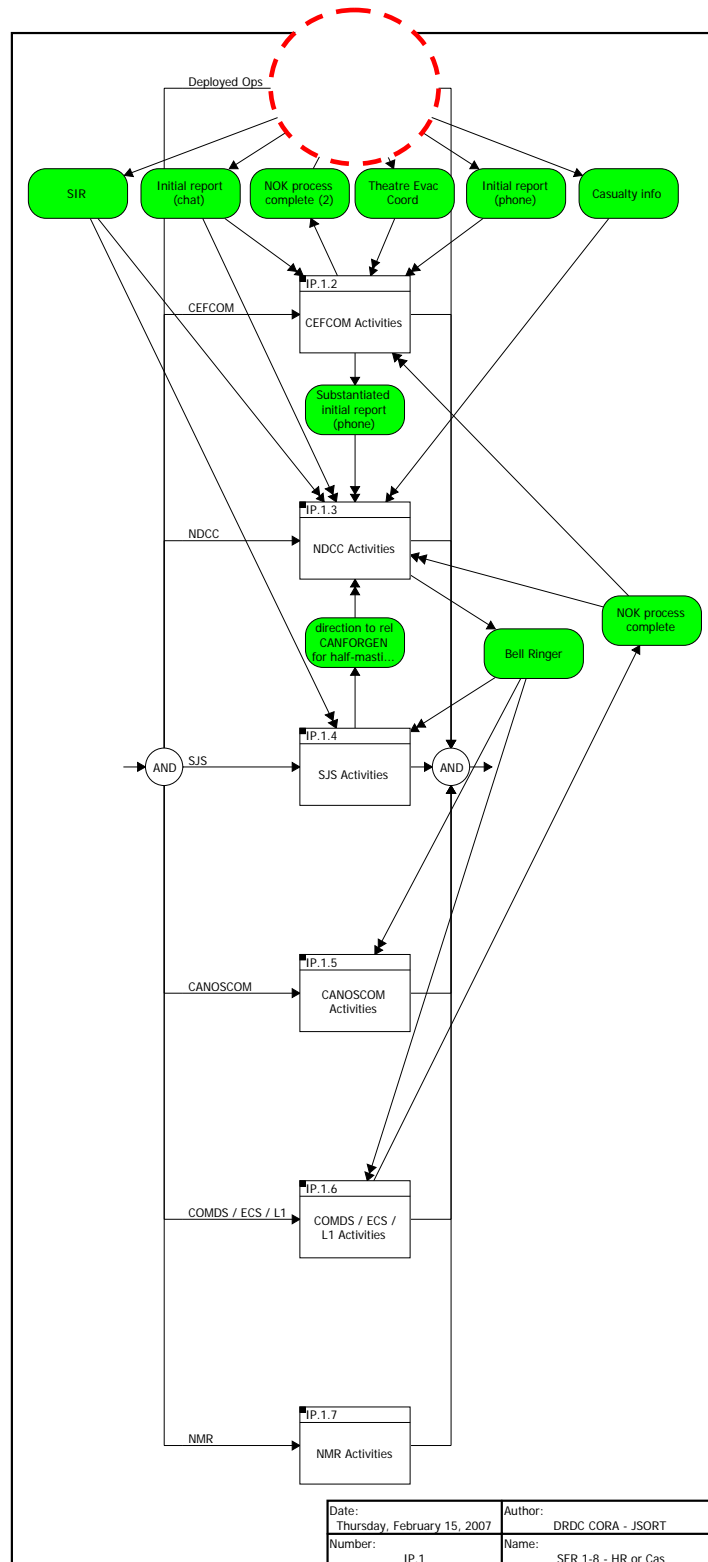


Figure 11: International Casualty Reporting Model Showing Unit Interactions in Serials 1-8

Figure 12 illustrates the activities undertaken by the Deployed Ops team in serials 1 through 8, represented at the higher level by the circled box in Figure 11. The dashed red lines and numbers show how the activities are split between serials (no activities are shown for Serial 8 as Deployed Ops only has responsibility to maintain SA in the SOP Serial 8). Note that some serials are done in parallel or even out of order (eg. the lockdown based on Serial 1 is only complete after Serial 2).

The white activity boxes in Figure 12 represent activities that existed in the original serial version of the model (note the IS in the numbering scheme, representing International – Serial). The yellow elements were added so the SOP logic would be more realistic and are unique to this version of the model (their numbering scheme contains IP for International – Parallel).

The process as modelled indicates that the procedure that operators actually follow works in a very different way than the existing SOP documentation would suggest. The full model is illustrated in Annex D.

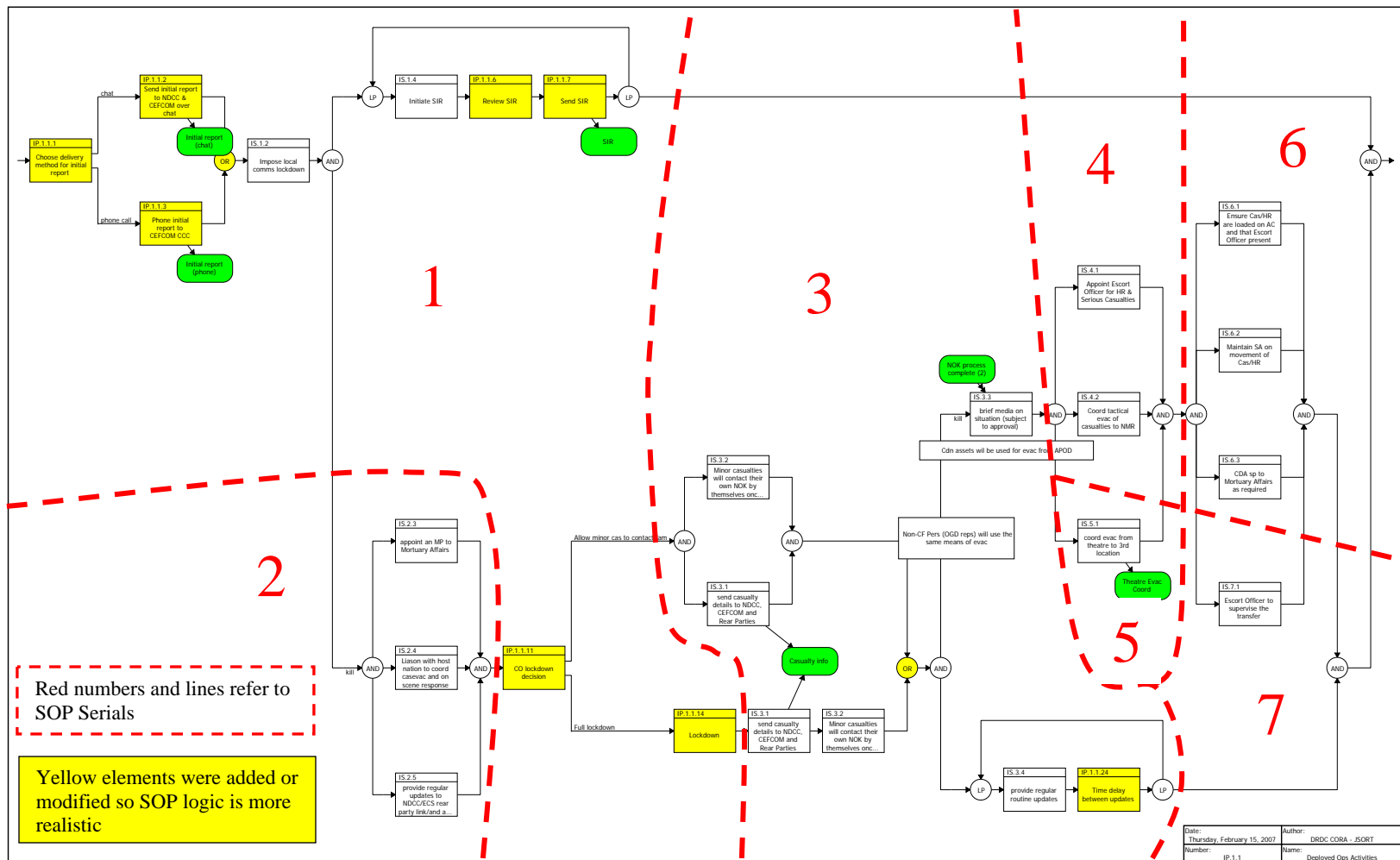


Figure 12: Casualty Reporting Within The Internationally Deployed Unit During Serials 1-8

6.3 Findings of the Trial

Both trial cases successfully demonstrated how to quickly build and validate detailed models with operator SMEs. The results were much better articulated C2 processes than existed before and they provided a means to test a broad range of process options. The added bonus of using CORE was that formatted DoDAF product documents could be automatically generated for whatever data was entered into the model. This gives the SMEs something to take away as a basis for documenting a revised SOP.

The trial results indicate that the process of reviewing a few key SOPs should take only a few days or weeks. The results of these trial cases were briefed to 7th Canadian Forces Operations Centre Conference (CFOCC) and were well received. Although feedback was positive, no commitments have been made by the Operational Commands to provide support for future work. However, follow-up by Joint Command Decision Support for the 21st Century (JCDS21) led to a collaborative effort to model further C2 operational architecture segments, including the Canada Command Rapid Response Action Planning process, the Combined Forces Air Component Command (CFACC) National Aerospace Planning Process (NAPP), and SOPs related to security at the Vancouver 2010 Olympic Games.

There are several SOPs that have been suggested as future candidates to be included in the SMOFN, on the grounds that the SMEs believed those processes could be improved if they were properly articulated and documented. These would be given the same treatment as the eBrief and Casualty Reporting processes. The list is presented in Annex E.

The greatest challenge in implementing SMOFN will come from the need to have assured access to operator SMEs. They are the key to collecting, collating and validating the full range of C2 process threads. Unfortunately, these same operator SMEs are also the resource in greatest demand for ongoing operations. The staffs are investigating how to support the review of their existing SOPs in order to document the C2 processes needed to support the future operational architecture.

The utility of C2 process modelling has been demonstrated. It is now for each Command to decide if they want to commit to the modelling of their SOPs. Those that participate will have to:

1. Submit a prioritized list of SOP threads to be modelled;
2. Assign SMEs to each SOP thread by name (SMEs must be available for several short sessions and should be same ones who will re-write SOPs);
3. Review SOP models as they are completed; and
4. Use the new SOP models as the basis for revised SOP documentation.

The final step in the modelling involves converting the C2 process threads to SMOFN data input file format. A portion of this has already been demonstrated using the above trial cases by decomposing the threads into simple sub-jobs and then linking them together. The manual conversion of threads to input files is a significant test of the completeness of the thread modelling, as many details of the processes are required by the SMOFN and the modelled thread

needs to be well understood by the person doing the conversion. The automation of this process is left for a future project to resolve.

6.4 Description of Proposed Approach

Based on the experience of the two modelled SOPs, JSORT recommends the steps outlined below as the method to collect further data. This process assumes the SMEs are familiar with the basic characteristics of Enhanced Functional Flow Block Diagrams (EFFBDs) in CORE. This is either accomplished by a prior exposure to the same style of diagrams or having the analyst brief a short introductory example. The method that JSORT proposes to follow when modelling SOPs and C2 Practices is as follows:

1. If SOP documentation is available, the analysts build an initial model based on that documentation. This allows them to learn about the topic so they can better lead discussions, and it also provides a starting point from which to build the true process model, saving a significant amount of the SMEs' time as they do not need to be present to build a model from scratch. If no previously documented SOP is available, the entire model must be populated based on SME descriptions of their work so extra interview sessions will be required;
2. Analysts conduct several short sessions with SMEs to present, and receive feedback on, the model as built. Operators describe what they do in order to expand the SOP or correct errors;
3. Once the process diagrams are considered accurate, SMEs provide estimated values such as the amount of time or number of people required to complete various activities;
4. Analysts run the simulation of the modelled process to confirm that it executes as expected, and fix any issues that are identified in the process;
5. The steps involving SME input can be repeated with other SMEs in order to focus on different aspects within the model;
6. Analysts generate a report that operators can use to revise their SOP documentation, and which the analysts keep a copy of to adapt as input to the SMOFN.

7 Conclusions and Recommendations

A few significant conclusions have been drawn from the SMOFN modelling experience, as described in detail in this paper.

The SMOFN has achieved its goal of extending the detailed OPCEN SM model logic across multiple nodes, each with their own jobs and resources but also with the ability to share information. This information sharing takes place through the net-centric virtual Repository, whose operational logic successfully controls the interaction of information between the various operator perspectives (i.e. producer, consumer and discovery) as well as between different OPCENs within the SMOFN.

Typical OV-6 diagrams and other simulations as usually employed are limited in scope and flexibility by only being able to model what is built directly into them. On the other hand, SMOFN reads the desired organizational structure and operational logic from data files each time the simulation is started, allowing it to be both scalable and flexible. This also allows any details of the SMOFN itself to remain unclassified, even though the files that it uses at input may contain classified data.

The process of gathering data for process threads has been demonstrated to be both quick and effective, requiring a few days or weeks to produce a functioning, validated model. The major limitation noted so far is the availability of SMEs to provide the operational context, and no SME hours have been committed by the operational commands for future work. This is expected to continue to be the biggest implementation hurdle to building faithful simulation and executable operational architectures. On the other hand, the SME interaction has produced immediate results (in the form of DoDAF/DNDAF-compatible products) that can be used by the SMEs' home organizations as the basis for writing proper SOPs.

Our paper on the OPCEN SM [7] included a discussion of the metrics, such as variations in the job completion rate, that can be used to evaluate different model configurations (ie. sets of input files). The output files of the SMOFN, like the OPCEN SM, contain an account of the state of every active job and operator at every time step, the only addition being the specification of the OPCEN where those jobs and operators are located. The same metrics that were available to the OPCEN SM are therefore still available to the SMOFN, and are still relevant on its larger scale.

The SMOFN engine is a viable simulation of OPCEN operations and interactions. Although a large amount of data will be required to make the model reflective of real processes, JSORT's recommendation is that the SMOFN should be used as a testbed for potential changes to operational procedures. To make the input data as manageable as possible, it may be possible to create a program that converts threads modelled in CORE to the input files required by the SMOFN. This may be facilitated by the fact that CORE is able to export all information pertinent to the threads as xml. To make the simulation and its results as realistic as possible, it is also recommended that the necessary SME hours be made available as a precondition for modelling processes to any degree of fidelity. This investment of time would make SMOFN a valuable tool in studying how the CF does, and potentially could, operate.

References

- [1] Wikipedia, the free encyclopedia, Federated Database System, accessed 16 November 2009, http://en.wikipedia.org/wiki/Federated_database_system
- [2] Stenbit, J.P. *DoD Architecture Framework (DoDAF) Version 1.0* (http://www.defenselink.mil/nii/doc/DoDAF_v1_Memo.pdf), 9 February 2004. Composed of two volumes and a deskbook:
 - DoDAF Version 1.0 - Volume I, 9 February 2004 (www.defenselink.mil/nii/doc/DoDAF_v1_Volume_I.pdf)
 - DoDAF Version 1.0 - Volume II, 9 February 2004 (www.defenselink.mil/nii/doc/DoDAF_v1_Volume_II.pdf)
 - DoDAF Version 1.0 – Deskbook, 9 February 2004 (www.defenselink.mil/nii/doc/DoDAF_v1_Deskbook.pdf)
- [3] Canadian Forces C4ISR Command Guidance and Campaign Plan, 2 December 2003 <http://cdcs.mil.ca/dgjfd/djfc/C4ISR/documents/2150610547PM20031202%20Command%20Guidance%20&%20Campaign%20Plan%20-%20main.doc>
- [4] SM Definition, accessed 5 June 2008, http://searchsmallbizit.techtarget.com/sDefinition/0,,sid44_gci214244,00.html
- [5] Funk, R.W., Ball, M.G. and Sorensen, R. “Building Executable Architectures of Net Enabled Operations Using State Machines to Simulate Concurrent Activities”, paper I-014 presented to 11th ICCRTS, Cambridge, UK, 28 September 2006. http://www.dodccrp.org/events/11th_ICCRTS/html/papers/014.pdf
- [6] Funk, R.W. and Sorensen, R.L., “State Machine Modelling of TPED and TPPU”, NDIA Proceedings, October 2005. <http://www.dtic.mil/ndia/2005systems/thursday/sorensen.pdf>
- [7] Funk, R.W., Ball, M.G. and Sorensen, R.L., Building Executable Architectures to Simulate Concurrent Activities of Net Enabled Operations - Case of a Single Operations Centre, DRDC Technical Report TR 2006-24, November 2006.
- [8] Gauthier, S.M. and Funk, R.W., Estimating Partial Utility of Interim Products, DRDC CORA Technical Note TN 2006-02, April 2006.
- [9] Sorensen, R.L., Funk, R.W. and Ball, M.G., “Exploring Concurrent Activities: Using State Machines to Understand Net-Enabled Operations”, paper P1111 presented to Seventeenth Annual International Symposium of the International Council On Systems Engineering (INCOSE) San Diego, CA, USA, 27 June 2007.

- [10] Ball, M.G., Funk, R.W., and Sorensen, R.L., “Executable Architecture of Net Enabled Operations: State Machine of Federated Nodes”, paper I-032 presented to 12th ICCRTS, Newport, RI, USA, 20 June 2007.
http://www.dodccrp.org/events/12th_ICCRTS/CD/html/papers/032.pdf
- [11] DND/CF Architecture Framework (DNDAF) v1.6, accessed 15 July 2009,
http://img.mil.ca/stratctr/architect/dndaf_e.asp
- [12] Alberts, D.S. and Hayes, J.L., Understanding Command and Control, DoD Command and Control Research, Program, Washington D.C., ISBN 1-893723-17-8, 2006.
http://www.dodccrp.org/publications/pdf/Alberts_UC2.pdf
- [13] Smith, P.A. and Carson, N.C., Marine Security Operation Centre (MSOC) East: Work and Information Flow Analysis: Phase 1 Report, DRDC CORA Technical Report TR 2006-11, October 2006, UNCLASSIFIED (GOVERNMENT USE ONLY).

This page intentionally left blank.

Annex A SMOFN User Manual

Each phase of the modelling contract included the production of a user manual providing instructions on installing and running the SMOFN. The latest version is included here for reference. It is divided into two main sections: the initial setup of the SMOFN engine, and the scenario-specific simulation.

A.1 Setup

This section describes the initial setup of the SMOFN. This requires the files “Scripts.zip”, “Initialization.zip”, and “SMOFN.xml” which can be provided electronically. Note that this manual assumes the typical installation of CORE such that the installation directory is “C:\Program Files\Vitech Corporation\CORE Workstation 50”. Also note that in Windows, any instances of “%username%” below will be automatically replaced with the name of the user currently logged in.

A.1.1 Install the scripts:

The SMOFN engine uses external scripts to calculate the values for product utility and utility decay. These scripts must be installed on the user’s computer to execute the SMOFN engine. Install the scripts to the CORE Reports\50 Reports\User\SM folder as follows:

1) Copy the "scripts.zip" file to folder:

C:\Program Files\Vitech Corporation\CORE Workstation 50\Reports\50 Reports\User

2) Once there, unzip. The scripts should now be installed in subfolder SM. The full path to the scripts is:

C:\Program Files\Vitech Corporation\CORE Workstation 50\Reports\50 Reports\User\SM

A.1.2 Install Initialization Files:

3) Copy the "Initialization.zip" file to your folder where you keep your input files, such as:

C:\Documents and Settings\%username%\My Documents\CORE\SMOFN

4) Once there, unzip. The files should now be installed in subfolder Initialization. The full path is:

C:\Documents and Settings\%username%\My Documents\CORE\SMOFN\Initialization

5) Fix the path in file S1-Datapath.txt to reflect your configuration.

A.1.3 Load the SMOFN project in CORE:

6) Locate the file SMOFN.xml and copy this file to the folder where you keep CORE exports, such as:

C:\Documents and Settings\%username%\My Documents\CORE

7) Start CORE and login.

8) In CORE's Project Explorer, go to File, Import

9) In the resulting Import File dialog box, navigate to the folder where you saved the file SMOFN.xml, select the file and click Open. The XML import wizard opens. Click Next twice.

NOTE: At Step 2, if you already have a project named SMOFN (ie. from a previous import), it is ***STRONGLY*** recommended that you create a new project rather than import into an existing project. At Step 3 of 3, click Import.

A.2 Simulation Procedure

Simulation execution consists of scenario development and subsequent model execution. Scenario development is the process of creating the set of input files required by the simulation. Although the files read by the SMOFN are tab-delimited text files (extension .txt), it has been the authors' practice to create and edit them as a spreadsheet and copy the final versions to text. This is often done using an Excel workbook with individual worksheets dedicated to each of the input files. Scenario development can also be accomplished by modifying a previous set of text input files, but the potential for file format errors makes it far more prudent to generate new files from the workbook.

The simulation requires a minimum of seventeen text files. The files are organized in three groups: A set of files that describe general scenario parameters, a set of files (which may be duplicated) that describe the aspects of one operations center, and a set of files that describe the repository. The files that describe scenario parameters are:

- S1-Datapath.txt
- S2-SimDuration.txt
- S3-DayTime.txt
- S4-Switchboard.txt
- S5-OpCenters.txt

The following five files articulate the characteristics and events for one operations center, and can be repeated for additional operations centers as appropriate. Because the file names are fixed, each individual operations center's file sets need to be located in separate directories.

- O1-ThreadTypes.txt
- O2-ThreadDefinitions.txt
- O3-Events.txt
- O4-SkillsMatrix.txt
- O5-DecayCurves.txt

The Repository folder, located beneath the primary scenario folder, contains the following files used to initialize Repository data delivery and to provide information on routing, bandwidth, and probabilities for Repository functions:

- R1-ExternalData.txt
- R2-ExternalProducts.txt
- R3-Delivery.txt
- R4-Bandwidth.txt
- R5-Questions.txt
- R6-DiscoveryRequests.txt
- R7-ExternalResponse.txt

A.2.1 Scenario Development – Scenario General Parameters

Scenario general parameters are those initialization parameters that are either applicable across the entire simulation, like the Switchboard entries, or are necessary for the initialization routines to establish file paths, time, and durations that are required to set up the simulation. There are five files in this group, and these need to be located in the root (the folder identified in the input file S1-Datpath.txt) folder for the scenario, with the exception of the file S1-Dtapath.txt itself. The convention is to put this file in the root folder also, but this is not required.

- S1-Datpath.txt
- S2-SimDuration.txt
- S3-DayTime.txt
- S4-Switchboard.txt
- S5-OpCenters.txt

S1-Datpath.txt

- The file S1-Datpath.txt consists of a single entry with the full path to the remainder of the initialization files, as in the following example:
 - ♦ C:\Documents and Settings\%username%\My Documents\CORE\SMOFN\Initialization\
 - ♦ Ensure that the entry includes the trailing ‘\’ character and nothing beyond.
- The file S1-Datpath.txt can be located wherever the user prefers, but the remaining general properties files must be located at the path specified.
- The file structure from this example is shown in Figure 13.

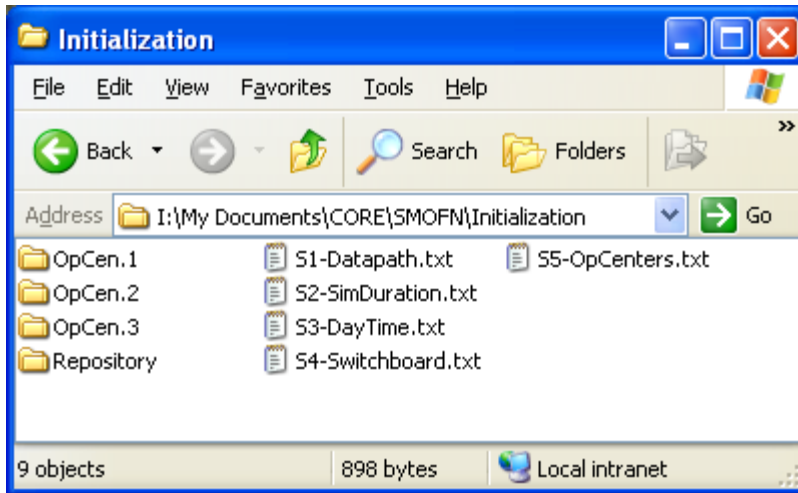


Figure 13: Folder structure example

- Suggestion: keep S1-Datapath.txt and the remaining initialization files collocated so that all associated files for any given scenario are aggregated for further analysis and historical record keeping.
- Results files will be placed in the same folder as the initialization files.

S2-SimDuration.txt

- The file S2-SimDuration.txt is a single entry file with simulation duration time in the format: days:hours:minutes. For example, 00:03:00 results in a simulation that ends 0 days, 3 hours, 0 minutes after starting (simulation time).

NOTE: The simulation may complete prior to the time entered in S2-SimDuration.txt. This will occur if there are no remaining inputs in the schedules for any of the operations centers.

S3-DayTime.txt

- The file S3-DayTime.txt, show in Figure 14, contains two entries. The first is a day of the week (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday). The second is clock time (00:00 through 23:59). In the example below, the scenario starts at 23:55 on Wednesday.

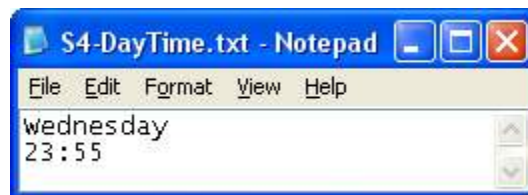
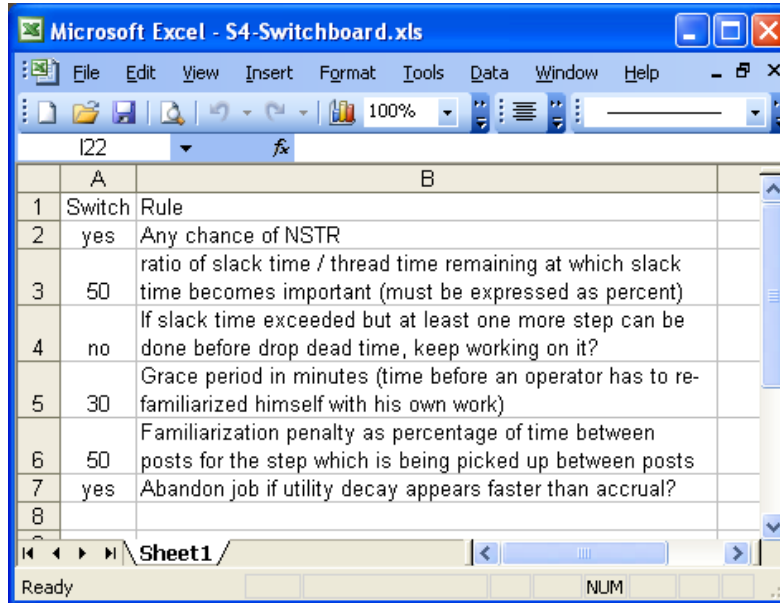


Figure 14: Simulation Start Time Example

S4-Switchboard

- The file S4-Switchboard.txt, shown in Figure 15, provides control features for specific aspects during the simulation execution
- See column B definitions for each item:



	A	B
1	Switch	Rule
2	yes	Any chance of NSTR
3	50	ratio of slack time / thread time remaining at which slack time becomes important (must be expressed as percent)
4	no	If slack time exceeded but at least one more step can be done before drop dead time, keep working on it?
5	30	Grace period in minutes (time before an operator has to re-familiarized himself with his own work)
6	50	Familiarization penalty as percentage of time between posts for the step which is being picked up between posts
7	yes	Abandon job if utility decay appears faster than accrual?
8		

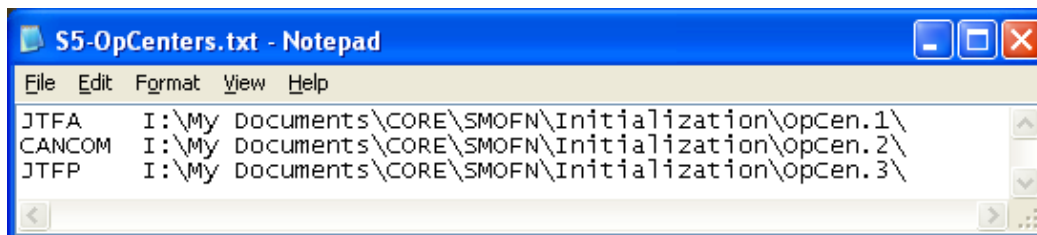
Figure 15: S4-Switchboard Example

- When saved, retain the header row but ensure there are no blank lines after the last line of text

Save with the file name: S4-Switchboard.txt

S5-OpCenters.txt

- The file S5-OpCenters.txt, shown in Figure 16, contains a single line entry for each operations center included in the scenario. Each entry is composed of two parts: the name of the OPCEN and the path to the folder containing the initialization details for that OPCEN.



```

JTFA I:\My Documents\CORE\SMOFN\Initialization\OpCen.1\
CANCOM I:\My Documents\CORE\SMOFN\Initialization\OpCen.2\
JTFF I:\My Documents\CORE\SMOFN\Initialization\OpCen.3\

```

Figure 16: S5-OpCenters.txt Example

The above S5-OpCenters.txt file reflects the folder structure in Figure 13. The OPCEN-specific initialization files are in each of the subordinate folders (i.e. OpCen.1). The contents of the OpCen.1 folder are shown in Figure 17. Although the data would likely be different, the file names under any OPCEN folder would be the same as in this example.

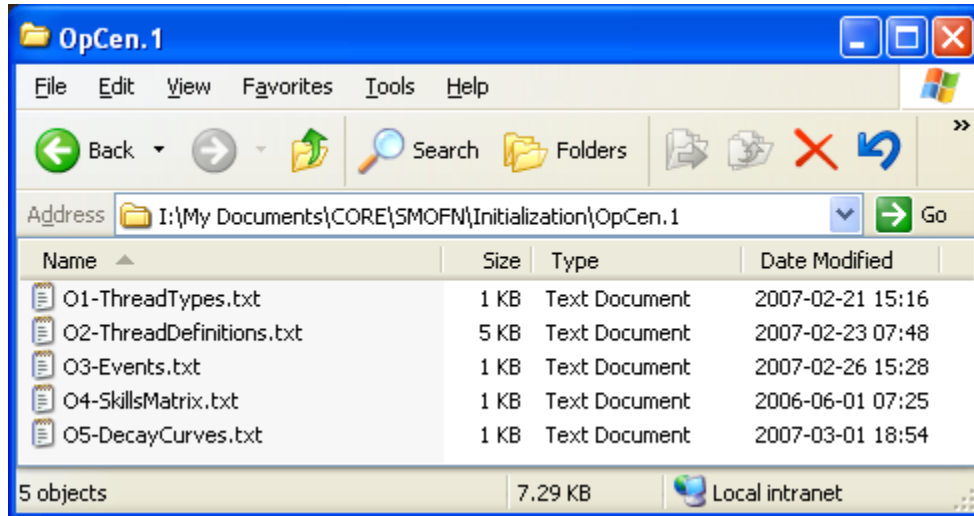


Figure 17: OPCEN 1 Folder Contents

Snapshot Interval

- This value represents the number of simulation minutes between snapshots (the contents of what will be written as Results)
- To change it, open the SMOFN project in CORE and locate the Resource named Snapshot Interval. Its Property Sheet window is reproduced in Figure 18. Set the Initial Amount to the desired interval
 - ♦ Smaller intervals result in more detail, larger output files.
 - ♦ Larger numbers result in less detail and smaller output files.

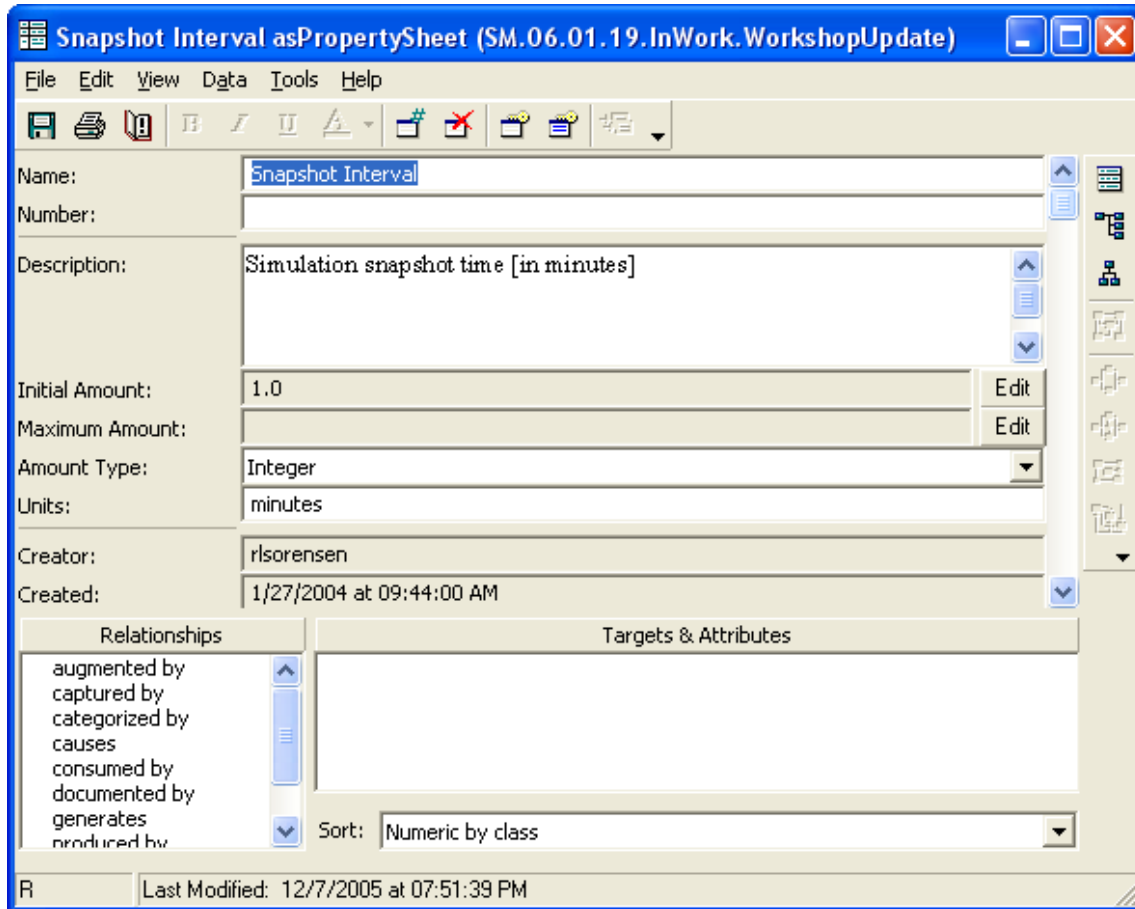


Figure 18: Setting the Snapshot Interval

A.2.2 Scenario Development – Operations Centers

Each Operations Center requires a set of input files to establish the specific parameters for that center. These are:

- O1-ThreadTypes.txt
- O2-ThreadDefinitions.txt

- O3-Events.txt
- O4-SkillsMatrix.txt
- O5-DecayCurves.txt

These files must be collected within one folder as shown in Figure 17.

Multiple Operations Centers for a given scenario can be located anywhere, but the recommended approach is to collect all the referenced Operations Centers, each in a separate folder, under the scenario root folder, as shown in Figure 13 and Figure 16.

O1-ThreadTypes

- The file O1-ThreadTypes.txt, shown in Figure 19, describes the nominal thread attributes used by the simulation
- Entries are in the format:
 - ♦ Thread Name, Standard Priorities, Fraction NSTR, Slack Time, Look Ahead, Redundancy, NSTR utility (%), Step where other jobs are incorporated, Job incorporation details, where:
 - Thread Name identifies the basic job thread:
 - Standard Priorities identifies the priority of the basic job thread
 - Fraction NSTR identifies the percentage of input events (by thread type and priority) which will be treated as having Nothing Significant To Report (NSTR)
 - Slack Time is the initial excess time allowed before work on a given product is halted, and is expressed in minutes
 - Look Ahead: The integer number of steps to look forward when determining potential utility to be gained. Used in determining when to abandon jobs.
 - Redundancy: either a yes or no value is entered. Used to determine if subsequent events of the same type occurring while an existing thread is in process are ignored.
 - NSTR utility (%): Integer value representing the percent utility of knowing that there is Nothing Significant To Report, should that be the case.
 - Step where other jobs are incorporated: Defines the step of the specific thread where other products are incorporated. The value 'nil' is used when there is no incorporation taking place within the thread. NOTE: This step name must correspond with a step name defined in the input file O2-ThreadDefinitions.txt
 - Job incorporation details: A list of the thread types to be incorporated. The list is composed of three attributes for each thread type incorporated: Thread name (including priority), number of minutes added, and percent contribution. Multiple thread types are allowed, separated by tabs.

Microsoft Excel - OPCEN Init 20090226.xls

File Edit View Insert Format Tools Data Window Help

Type a question for help

AA15

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	Thread Names	Standard Priorities	Fraction NSTR	Slack Time (Initial)	Look ahead	Redundant?	NSTR utility (%)	Step where other jobs incorporated	Incorporates, adds time, % contribution; incorporate					
2	Imagery	200	50%	30	3	no	90	nil						
3	Imagery	100	20%	20	3	no	50	nil						
4	Snapshot	400	10%	15	0	no	10	nil						
5	SITREP	300	5%	30	4	yes	50	Step_3	Imagery200	2	100	Snapshot400	1	100
6	SITREP	100	5%	15	4	yes	50	Step_3	Imagery100	2	100	Snapshot400	1	100
7	Assessment	200	33%	15	1	yes	30	Step_3	Imagery200	0.5	100	Imagery100	0.5	100
8	Assessment	100	10%	5	2	yes	30	Step_3	Imagery100	0.5	100	Snapshot400	0.5	100
9	Discovery	100	5%	30	0	yes	10	nil						
10														

O1-ThreadTypes / O2-ThreadDefinitions / O3-Events

Ready

NUM

Figure 19: O1-ThreadTypes Example

- When saved, retain the header row but ensure there are no blank lines after the last line of text
- Save with the file name: O1-ThreadTypes.txt

O2-ThreadDefinitions

- The file O2-ThreadDefinitions.txt, shown in Figure 20, contains the definitions of each step in each job thread as required for interpretation by the SMOFN.
- Entries are in the format:
 - ♦ Thread Name, Priority, Step Name, Duration, Skill, Posts, Interrupts, lowerBound, Peak, upperBound, Step Type, Next Step, where:
 - Thread Name identifies the basic job thread
 - Priority identifies the priority of the basic job thread
 - Step Name: identifies the step within the thread. These names will be reflected in the results files and should reflect the activity of the step performed. The one exception is that a thread definition begins with the step name START. Step names must be unique within each combination of Thread Name and Priority.
 - Duration: an integer value representing the nominal duration of the step in minutes.
 - Skill: the skill required to perform the step. The skill name should be “G” for generic, or consistent with one of the Skills identified in input file O4-SkillsMatrix, otherwise no operator can be assigned to the job step.

- Posts: an integer number representing the number of times information is posted to the repository during the step.
- Interrupts: indicates whether the step is interruptible. Enter either a 0 (not interruptible) or a 1 (interruptible).
- lowerBound, Peak, upperBound: three integer values defining the triangular distribution of utility accrued by the end of the associated step. For each step the values must conform to the expression: $\text{lowerBound} \leq \text{Peak} \leq \text{upperBound}$.
- Product Size: an integer value representing the file size of the final product uploaded to the Repository.
- Step Type: Currently seven step types are defined. All normal thread steps are step Type 1. Step Type 2 defines an NSTR decision point. Step Type 3 is the final step of a production job. Step Types 4 through 7 handle the results of a discovery thread, as described in Section 4.2.2. These should normally only appear as shown below in Figure 21.
- Next Step: the name of the next step in the thread. Step names entered here must appear in the Step Name column. The last step in a thread sequence must show the Next Step entry as COMPLETE.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Thread Name	Priority	Step Name	Duration	Skill	Posts	Interrupts	lowerBound	Peak	upperBound	Product Size	Step Type	Next Step
1	Imagery	100	START	0	G	1	0	1	2	3	10	1	Step_1
2	Imagery	100	Step_1	5	IA	1	0	5	9	15	10	1	Step_2
3	Imagery	100	Step_2	1	G	1	1	10	19	25	20	2	Step_3
4	Imagery	100	Step_3	5	IA	3	0	20	29	35	30	1	Step_4
5	Imagery	100	Step_4	5	IA	2	0	30	39	45	40	1	Step_5
6	Imagery	100	Step_5	5	CM	1	1	40	49	55	50	1	Step_6
7	Imagery	100	Step_6	1	IA	1	1	50	59	65	60	1	Step_7
8	Imagery	100	Step_7	4	IA	2	1	60	69	75	70	1	Step_8
9	Imagery	100	Step_8	6	IA	3	1	70	79	85	80	1	Step_9
10	Imagery	100	Step_9	2	CM	1	1	80	95	100	90	1	Step_10
11	Imagery	100	Step_10	0	G	0	1	0	0	0	90	3	COMPLETE
12	Imagery	200	START	0	G	1	0	1	2	3	10	1	Step_1
13	Imagery	200	Step_1	5	IA	1	0	5	9	15	10	1	Step_2
14	Imagery	200	Step_2	1	G	1	1	10	19	25	20	2	Step_3
15	Imagery	200	Step_3	5	IA	2	1	20	29	35	30	1	Step_4

Figure 20: O2-ThreadDefinitions Example

- When saved, retain the header row but ensure there are no blank lines after the last line of text
- Save with the file name: O2-ThreadDefinitions.txt

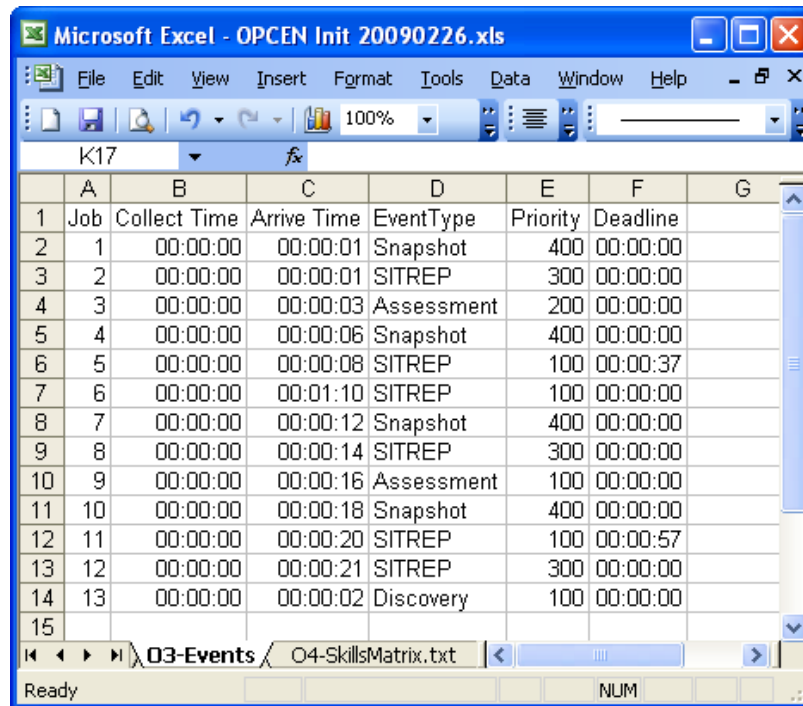
NOTE: The Discovery thread is a special case of a defined thread, as shown in Figure 21. Do not alter the highlighted entries without first understanding the interaction between these steps and the implementation in Thread & Queue Processing in the CORE SMOFN engine.

Thread Name	Priority	Step Name	Duration	Skill	Posts	Interrupts	lowerBound	Peak	upperBound	Product Size	Step Type	Next Step
Discovery	100	START	0	G	1	0	1	2	3	10	1	Receive_Consumer
Discovery	100	Receive_Consumer_Question	1	SWO	1	0	5	5	5	10	1	Review_Question
Discovery	100	Review_Question	2	SWO	1	1	5	10	10	10	1	Provide_Direction
Discovery	100	Provide_Direction	5	SWO	1	0	10	15	20	10	1	CM_Review
Discovery	100	CM_Review	2	CM	1	1	10	15	20	10	1	Formulate_Query
Discovery	100	Formulate_Query	5	CM	1	1	15	20	25	10	1	Query_Rep
Discovery	100	Query_Rep	3	CM	1	0	26	26	26	10	1	Rep_Results_Decis
Discovery	100	Rep_Results_Decision	10	CM	1	0	90	100	100	10	1	Rep_Results_Action
Discovery	100	Rep_Results_Action	2	CM	1	0	90	100	100	10	4	COMPLETE
Discovery	100	Formulate_New_Delivery	19	CM	1	0	90	100	100	10	1	Delivery
Discovery	100	Delivery	1	CM	1	0	90	100	100	10	5	COMPLETE
Discovery	100	Identify_Repository_Items	4	CM	1	0	90	100	100	10	1	Insert_New_Job
Discovery	100	Insert_New_Job	1	CM	1	0	90	100	100	10	6	COMPLETE
Discovery	100	New_Delivery_plus_New_Job	23	CM	2	0	90	100	100	10	1	Delivery_plus_New
Discovery	100	Delivery_plus_New_Job	1	CM	1	0	90	100	100	10	7	COMPLETE

Figure 21: O2-ThreadDefinitions with Discovery Thread Highlighted

O3-Events

- The file O3-Events.txt, shown in Figure 22, is a listing of the input events for the simulation
- Entries are in the format:
 - ♦ Job Number, Collect and Arrive Time, Event Type, Priority, and Deadline, where:
 - Job Number: must be sequential, starting at 1
 - Collect Time: in the format days:hours:minutes from the start of the simulation. This represents the time at which the raw data was collected, which is used to determine the utility decay due to aging
 - Arrive Time: in the format days:hours:minutes from the start of the simulation until this job is added to the OPCEN queue
 - Event Type: identifies the basic job thread
 - Priority: identifies the priority of the job. For the purpose of cross-referencing to the other input files, the priority will be rounded to the nearest value corresponding to a basic job thread
 - Deadline: in the format days:hours:minutes, used to override (if desired) the default slack time for a specific event process. Enter 00:00:00 to use standard slack time



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - OPCEN Init 20090226.xls". The spreadsheet displays a table with 7 columns: Job, Collect Time, Arrive Time, EventType, Priority, and Deadline. The data is organized into 15 rows, with the first row serving as a header. The table content is as follows:

	A	B	C	D	E	F	G
	Job	Collect Time	Arrive Time	EventType	Priority	Deadline	
1	1	00:00:00	00:00:01	Snapshot	400	00:00:00	
2	2	00:00:00	00:00:01	SITREP	300	00:00:00	
3	3	00:00:00	00:00:03	Assessment	200	00:00:00	
4	4	00:00:00	00:00:06	Snapshot	400	00:00:00	
5	5	00:00:00	00:00:08	SITREP	100	00:00:37	
6	6	00:00:00	00:01:10	SITREP	100	00:00:00	
7	7	00:00:00	00:00:12	Snapshot	400	00:00:00	
8	8	00:00:00	00:00:14	SITREP	300	00:00:00	
9	9	00:00:00	00:00:16	Assessment	100	00:00:00	
10	10	00:00:00	00:00:18	Snapshot	400	00:00:00	
11	11	00:00:00	00:00:20	SITREP	100	00:00:57	
12	12	00:00:00	00:00:21	SITREP	300	00:00:00	
13	13	00:00:00	00:00:02	Discovery	100	00:00:00	
14							
15							

The Excel window also shows a taskbar at the bottom with "O3-Events" and "O4-SkillsMatrix.txt" as open files. The status bar at the bottom indicates "Ready" and "NUM".

Figure 22: O3-Events Example

- When saved, retain the header row but ensure there are no blank lines after the last line of text

- Save with the file name: O3-Events.txt

O4-SkillsMatrix

- The file O4-SkillsMatrix.txt, shown in Figure 23, identifies the operators available for tasking, their associated skills and performance, and their availability.
- Entries are in the format:
 - ♦ Operator Name, Skill Focus, Shift Hours, Shift Days, % Speed of Work, % Quality of Work, Order G Assigned, % G Available, Skill 1 through Skill 10
 - Operator Name: a name used to identify individual operators. All names must be unique within the file.
 - Skill Focus: the primary skill set of this operator.
 - Shift Hours: the number of hours this operator works in the assigned shift.
 - Shift Days: the number of days this operator works in a week.
 - % Speed of Work: how fast this operator works, with 100% being normal. Faster workers will have this value set higher; slower workers will have this value set lower.
 - % Quality of Work multiplier: used to modify the Utility value when a product is posted by this operator, based on their skill.
 - Order G Assigned: used to determine the order in which a generic (skill 'G') task is assigned to available operators. Enter a number between 1 and the number of operators. Each operator must be assigned a unique ordinal.
 - % G Available: used to identify the probability that the associated operator will be available for generic steps.
 - Skill 1 through Skill 10 are the skills in order of preference for a given operator.

Note that Skill Focus, Shift Hours, and Shift Days are currently unused in the simulation but are expected to be incorporated with further development.

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Name	Skill Focus	Shift Hours	Shift Days	% Speed of Work	% Quality of Work	Order G Assigned	%G Avail	Skill 1	Skill 2	Skill 3	Skill 4	Skill 5
1	Operator1	SWO	24	7	100	100	6	10	SWO	OI	nil	nil	nil
2	Operator2	IA	24	7	150	200	5	20	IA	SWO	CM	nil	nil
3	Operator3	IA	24	7	50	75	2	25	IA	CM	nil	nil	nil
4	Operator4	OI	24	7	75	150	3	50	OI	CM	nil	nil	nil
5	Operator5	CM	16	7	200	90	4	75	CM	IA	nil	nil	nil
6	Operator6	SS	8	5	25	100	1	99	SS	nil	nil	nil	nil

Figure 23: O4-SkillsMatrix Example

- When saved, retain the header row but ensure there are no blank lines after the last line of text
- Save with the file name: O4-SkillsMatrix.txt

O5-DecayCurves

- The file O5-DecayCurves.txt, shown in Figure 24, identifies the stepwise decay of product utility over time as a set of times when the decay is applied and the triangular distribution of the remaining utility at that time.
- Entries are in the format:
 - ♦ ProductName, Time, lowerBound, Peak, upperBound
 - ProductName: identifies the basic job thread and must be one of those defined in input file O2-ThreadDefinitions
 - Time: an integer value, measured from the time of observation (see Collect Time in file O3-Events), when the decay occurs
 - lowerBound: an integer value between 0-100 which identifies the lower bound of a triangular distribution
 - Peak: an integer value between 0-100 which identifies the mode, or peak probability, of a triangular distribution
 - upperBound: an integer value between 0-100 which identifies the upper bound of a triangular distribution

NOTE: For each entry in this file, the values of lowerBound, Peak, and upperBound must conform to the expression $\text{lowerBound} \leq \text{Peak} \leq \text{upperBound}$.

Microsoft Excel - OPCEN Init 20090226.xls

	A	B	C	D	E	F
1	ProductName	Time	lowerBound	Peak	upperBound	
2	Imagery100	0	50	75	100	
3	Imagery100	20	10	25	50	
4	Imagery100	45	0	0	10	
5	Imagery200	30	75	90	100	
6	Imagery200	60	10	60	75	
7	Imagery200	90	0	0	10	
8	Snapshot400	0	90	90	100	
9	Snapshot400	1	10	20	90	
10	Snapshot400	30	0	0	10	
11	SITREP100	60	0	0	10	
12	SITREP300	60	70	75	100	
13	SITREP300	100	10	23	70	
14	SITREP300	120	0	0	10	
15	Assessment100	60	0	0	10	
16	Assessment200	60	70	80	100	
17	Assessment200	100	8	50	72	
18	Assessment200	120	0	0	10	
19	Discovery100	100	8	50	72	
20	Discovery100	120	0	0	10	
21						

Ready NUM

Figure 24: O5-DecayCurves Example

- When saved, retain the header row but ensure there are no blank lines after the last line of text
- Save with the file name: O5-DecayCurves.txt

A.2.3 Scenario Development – Repository

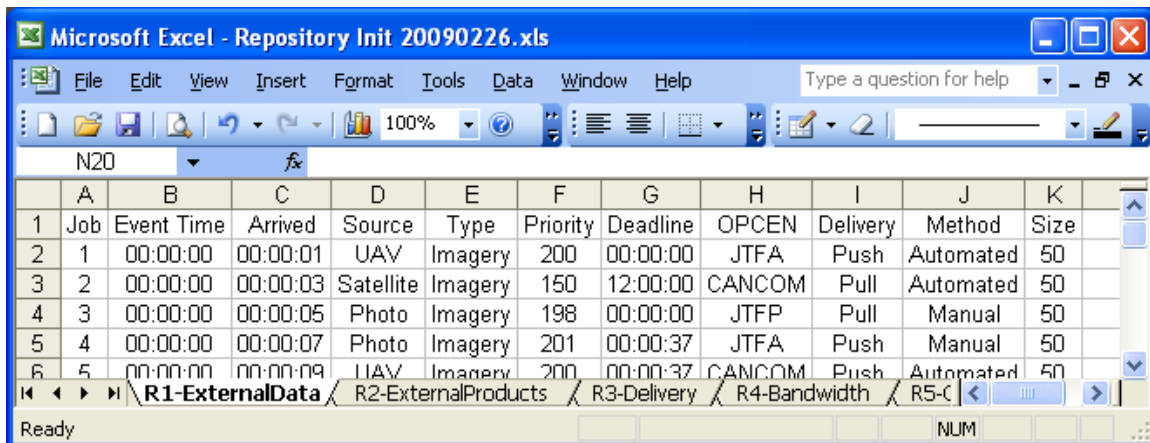
The Repository folder, located beneath the primary scenario folder, contains the following files used to initialize the Repository data delivery and to provide information on routing, bandwidth, and probabilities for Repository functions:

- R1-ExternalData.txt
- R2-ExternalProducts.txt
- R3-Delivery.txt
- R4-Bandwidth.txt
- R5-Questions.txt
- R6-DiscoveryRequests.txt

- R7-ExternalResponse.txt

R1-ExternalData

- The file R1-ExternalData.txt, shown in Figure 25, contains the initialization data for the delivery of external inputs
- Entries are in the format:
 - ♦ Job Number, Event Time, Arrived, Source, Type, Priority, Deadline, OPCEN, Delivery, Method, Size
 - Job Number: must be sequential, starting at 1
 - Event Time: the time at which the data to be analyzed was created (e.g. for an Imagery analysis job, the time at which the image was taken)
 - Arrived: the time at which the data will arrive at the Repository
 - Source: the source used to create the data. Currently unused by the simulation.
 - Type: the type of job thread that will be triggered by the arrival of this data
 - Priority: the priority of the job thread that will be triggered
 - Deadline: the time at which the job thread will be abandoned (00:00:00 indicates that standard slack time for the job type should be used to calculate the deadline)
 - OPCEN: the OPCEN that will perform the job thread
 - Delivery: indicates whether the Repository will deliver the data to the OPCEN via Push or Pull
 - Method: whether the Push or Pull is Automated or Manual
 - Size: the file size of the data that must be delivered

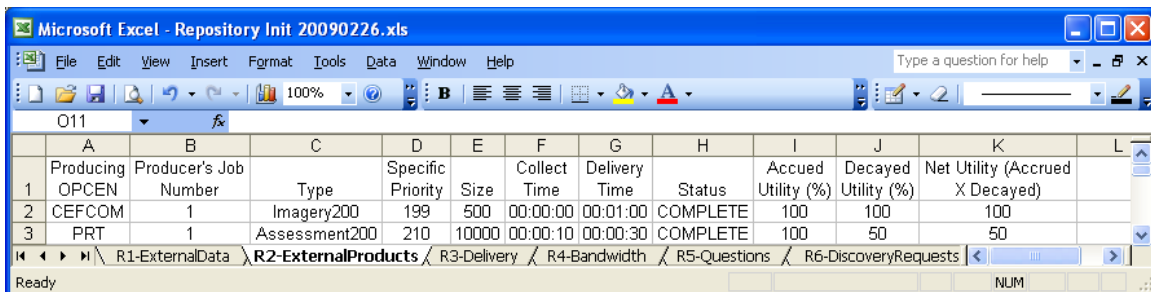


	A	B	C	D	E	F	G	H	I	J	K
	Job	Event Time	Arrived	Source	Type	Priority	Deadline	OPCEN	Delivery	Method	Size
1	1	00:00:00	00:00:01	UAV	Imagery	200	00:00:00	JTFA	Push	Automated	50
2	2	00:00:00	00:00:03	Satellite	Imagery	150	12:00:00	CANCOM	Pull	Automated	50
3	3	00:00:00	00:00:05	Photo	Imagery	198	00:00:00	JTFP	Pull	Manual	50
4	4	00:00:00	00:00:07	Photo	Imagery	201	00:00:37	JTFA	Push	Manual	50
5	5	00:00:00	00:00:09	UAV	Imagery	200	00:00:37	CANCOM	Push	Automated	50

Figure 25: R1-ExternalData example

R2-ExternalProducts

- The file R2-ExternalProducts.txt, shown in Figure 26, contains the initialization data for delivery of external products. These are the products created by OPCENs outside the scenario but intended to be used by the scenario.
- Entries are in the format:
 - ♦ Producing OPCEN, Producer's Job Number, Type, Specific Priority, Size, Collect Time, Delivery Time, Status, Accrued Utility, Decayed Utility, Net Utility
 - Producing OPCEN: the external OPCEN that created the product
 - Producer's Job Number: the job number used by the producing OPCEN to identify the specific product
 - Type: the type of job that resulted in the product, including its rounded priority
 - Specific Priority: the specific priority of the product
 - Size: the file size of the product
 - Collect Time: the time at which the raw data was collected
 - Delivery Time: the time at which the product will arrive at the Repository
 - Status: the state of completion of the product, identified by the name of the job step it has reached
 - Accrued Utility: the percentage of utility gained through progress on the job
 - Decayed Utility: the percentage of utility remaining due to the age of the product
 - Net Utility: the product of Accrued and Decayed Utility



	A	B	C	D	E	F	G	H	I	J	K	L
	Producing OPCEN	Producer's Job Number	Type	Specific Priority	Size	Collect Time	Delivery Time	Status	Accrued Utility (%)	Decayed Utility (%)	Net Utility (Accrued X Decayed)	
1	CEFCOM	1	Imagery200	199	500	00:00:00	00:01:00	COMPLETE	100	100	100	
2	PRT	1	Assessment200	210	10000	00:00:10	00:00:30	COMPLETE	100	50	50	

Figure 26: R2-ExternalProducts Example

R3-Delivery

- The file R3-Delivery.txt, shown in Figure 27, contains the initialization data for repository routing of products (whether created within the simulation or through R2-ExternalProducts).
- Entries are in the format:
 - ♦ Product, Producer Echelon, Producer, Consumer Echelon, Consumer, Probability, Delivery, Method
 - Product: the product type, including thread name and rounded priority

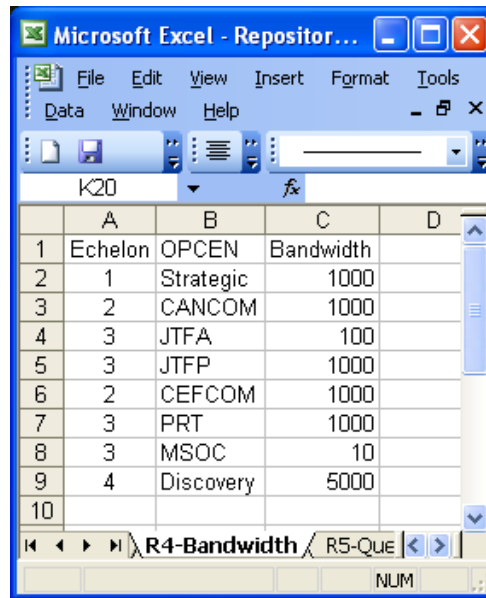
- Producer Echelon: the level of the Producer OPCEN within the command structure. This is used only for reference and is ignored by the simulation
- Producer: the OPCEN that may create the product
- Consumer: the OPCN that may receive the product when created by the Producer
- Consumer Echelon: as with the Producer, this indicates the level of the Consumer OPCEN within the command structure and is used only for reference.
- Probability: the percent chance that the Product will need to be delivered to the Consumer if created by the Producer
- Delivery and Method: as with the External Data, these identify whether the Product is delivered via Push or Pull, and whether that delivery will be Automated or Manual

	A	B	C	D	E	F	G	H
	Product	P.Echelon	Producer	C.Echelon	Consumer	Probability (%)	Delivery	Method
2	Snapshot400	2	CANCOM	1	Strategic	90	Pull	Automated
3	Snapshot400	2	CEFCOM	1	Strategic	80	Pull	Automated
4	Snapshot400	2	CEFCOM	3	PRT	80	Pull	Manual
5	Snapshot400	3	JTFA	2	CANCOM	30	Push	Manual
6	Snapshot400	3	JTFA	3	JTFP	50	Push	Automated
7	Snapshot400	3	JTFP	2	CANCOM	40	Push	Automated
8	Snapshot400	3	JTFP	3	JTFA	50	Pull	Manual
9	Snapshot400	3	PRT	2	CEFCOM	70	Push	Manual
10	Snapshot400	1	Strategic	2	CEFCOM	60	Pull	Automated
11	Imagery200	3	JTFA	3	JTFP	50	Push	Automated
12	Imagery200	3	JTFP	3	JTFA	50	Pull	Manual
13	Imagery100	3	JTFA	3	JTFP	50	Push	Manual
14	Imagery100	3	JTFP	3	JTFA	50	Pull	Automated
15	SITREP300	2	CANCOM	1	Strategic	90	Pull	Automated
16	SITREP300	2	CEFCOM	1	Strategic	80	Pull	Manual
17	SITREP300	2	CEFCOM	3	PRT	80	Pull	Manual
18	SITREP300	3	JTFA	2	CANCOM	30	Push	Automated
19	SITREP300	3	JTFA	3	JTFP	50	Push	Automated
20	SITREP300	3	JTFP	2	CANCOM	40	Push	Manual
21	SITREP300	3	JTFP	3	JTFA	50	Pull	Manual
22	SITREP300	3	PRT	2	CEFCOM	70	Push	Automated
23	SITREP300	1	Strategic	2	CEFCOM	60	Pull	Automated

Figure 27: R3-Delivery Example

R4-Bandwidth

- The file R4-Bandwidth.txt, shown in Figure 28, contains the initialization data for bandwidth available for repository routing of products.
- Entries are in the format:
 - ♦ Echelon, OPCEN, Bandwidth
 - Echelon: the level of the OPCEN within the command structure. Currently unused by the simulation.
 - OPCEN: identifies each OPCEN that is connected to the Repository in the simulation.
 - Bandwidth: the amount of data (using the same units as all file size entries in all other input files) that can be transferred between the OPCEN and the Repository during one time step (typically one minute).



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - Repositor...". The active sheet is "R4-Bandwidth". The data is organized in a table with columns A, B, and C. Column A is labeled "Echelon", Column B is labeled "OPCEN", and Column C is labeled "Bandwidth". The data rows are numbered 1 through 9. The values in the table are as follows:

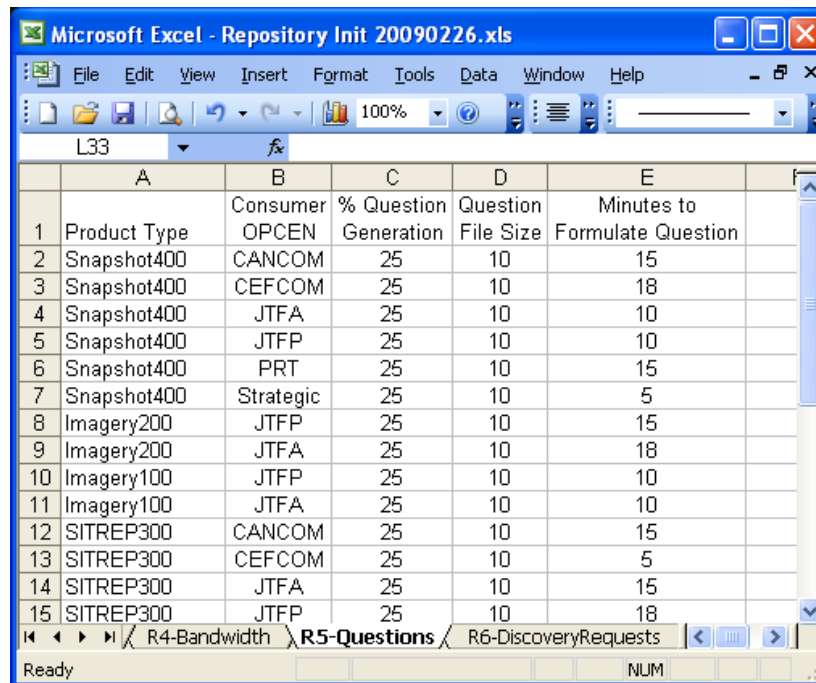
	A	B	C
1	Echelon	OPCEN	Bandwidth
2	1	Strategic	1000
3	2	CANCOM	1000
4	3	JTFA	100
5	3	JTFP	1000
6	2	CEFCOM	1000
7	3	PRT	1000
8	3	MSOC	10
9	4	Discovery	5000

Figure 28: R4-Bandwidth Example

R5-Questions

- The file R5-Questions.txt, shown in Figure 29, contains the initialization data for question generation by Consumers.
- Entries are in the format:
 - ♦ Product Type, Consumer OPCEN, % Question Generation, Question File Size, Minutes to Formulate Question
 - Product Type: identifies the product received by the consumer
 - Consumer OPCEN: identifies the OPCEN receiving the product

- % Question Generation: the probability that the product will cause a Question to be generated
- Question File Size: the size of the file containing the Question, sent to the Repository
- Minutes to Formulate Question: the delay time between the product being received and the Question being raised by the consumer



Microsoft Excel - Repository Init 20090226.xls

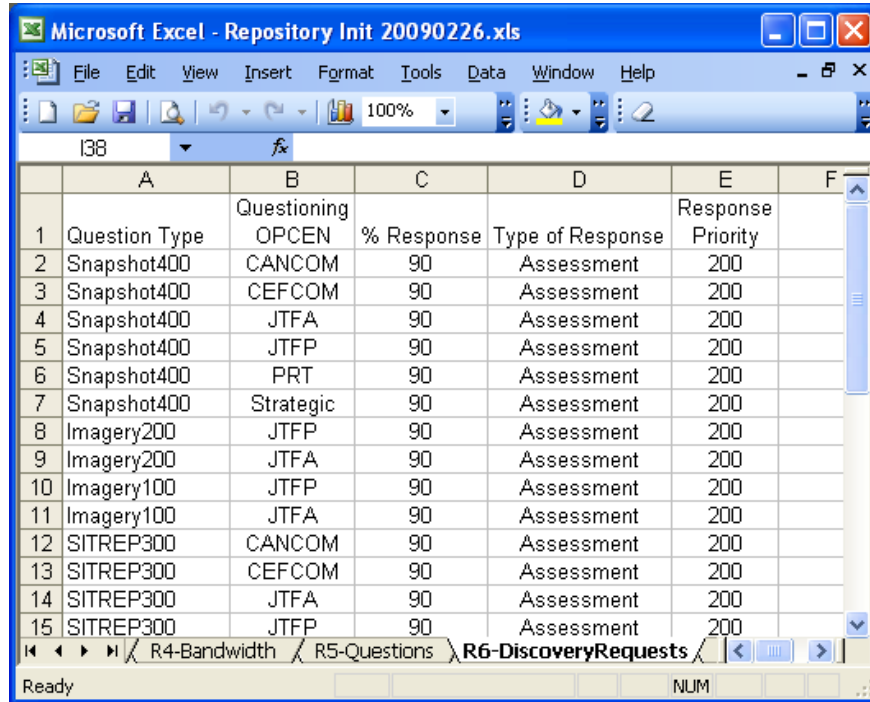
	A	B	C	D	E
	Product Type	Consumer	% Question Generation	Question File Size	Minutes to Formulate Question
1	Snapshot400	CANCOM	25	10	15
2	Snapshot400	CEFCOM	25	10	18
3	Snapshot400	JTFA	25	10	10
4	Snapshot400	JTFP	25	10	10
5	Snapshot400	PRT	25	10	15
6	Snapshot400	Strategic	25	10	5
7	Imagery200	JTFP	25	10	15
8	Imagery200	JTFA	25	10	18
9	Imagery100	JTFP	25	10	10
10	Imagery100	JTFA	25	10	10
11	SITREP300	CANCOM	25	10	15
12	SITREP300	CEFCOM	25	10	5
13	SITREP300	JTFA	25	10	15
14	SITREP300	JTFP	25	10	18
15	SITREP300	JTFP	25	10	18

Ready NUM

Figure 29: R5-Questions Example

R6-DiscoveryRequests

- The file R6-DiscoveryRequests.txt, shown in Figure 30, contains the initialization data for routing of questions from the Repository to Discovery. It defines the probability of receiving a response, and defines the response received.
- Entries are in the format:
 - ♦ Question Type, Questioning OPCEN, % Response, Response File Size, Minutes to Get Response to Repository, Type of Response, Response Priority
 - Question Type: identifies the type of product that the Consumer's question is based on
 - Questioning OPCEN: identifies the OPCEN that submitted the question
 - % Response: the probability that a question will be responded to
 - Type of Response: the basic job type to invoke upon receipt of the response by a Producer
 - Response Priority: the priority of the job to be invoked at the producer.



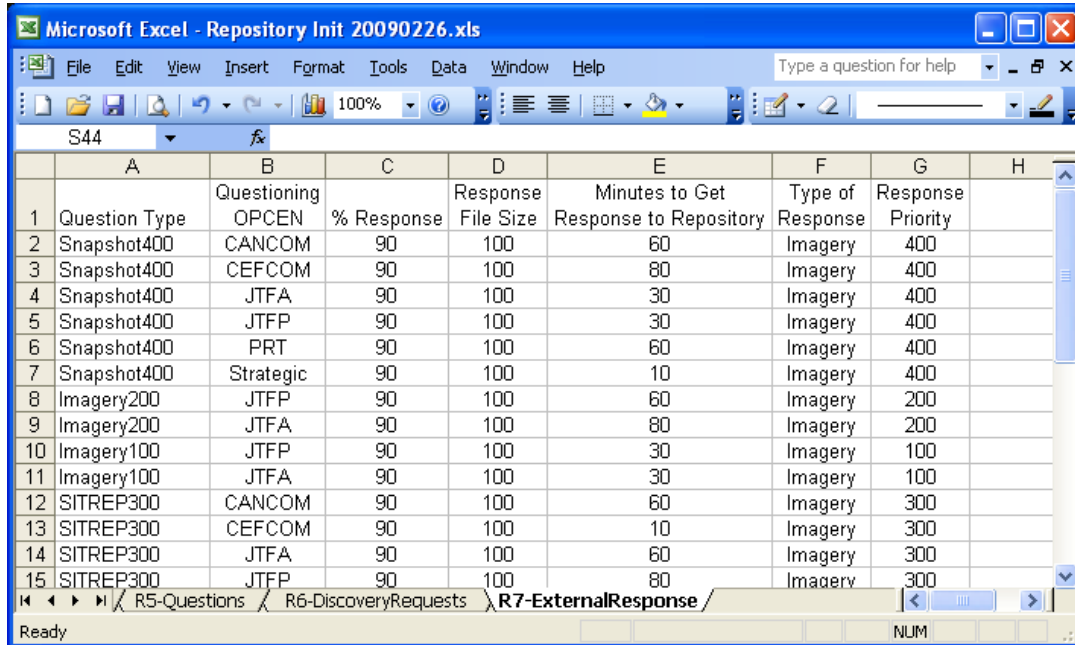
The screenshot shows a Microsoft Excel window titled "Repository Init 20090226.xls". The active sheet is "R6-DiscoveryRequests". The table contains the following data:

	A	B	C	D	E	F
	Question Type	Questioning OPCEN	% Response	Type of Response	Response Priority	
1	Snapshot400	CANCOM	90	Assessment	200	
2	Snapshot400	CEFCOM	90	Assessment	200	
3	Snapshot400	JTFA	90	Assessment	200	
4	Snapshot400	JTFP	90	Assessment	200	
5	Snapshot400	PRT	90	Assessment	200	
6	Snapshot400	Strategic	90	Assessment	200	
7	Imagery200	JTFP	90	Assessment	200	
8	Imagery200	JTFA	90	Assessment	200	
9	Imagery100	JTFP	90	Assessment	200	
10	Imagery100	JTFA	90	Assessment	200	
11	SITREP300	CANCOM	90	Assessment	200	
12	SITREP300	CEFCOM	90	Assessment	200	
13	SITREP300	JTFA	90	Assessment	200	
14	SITREP300	JTFP	90	Assessment	200	
15	SITREP300	JTFP	90	Assessment	200	

Figure 30: R6-DiscoveryRequests Example

R7-ExternalResponse

- The file R7-ExternalResponse.txt, shown in Figure 31, contains the same information as R6-DiscoveryRequests, except that instead of being completed from the perspective of questions being sent to Discovery, it is done from the perspective of queries sent to External Sources. It also adds two new entries between % Response and Type of Response:
 - ♦ Response File Size: the size of the file containing the response
 - ♦ Minutes to Get Response to Repository: the delay between receipt of the question by Discovery and receipt of a response at the Repository



	A	B	C	D	E	F	G	H
	Question Type	Questioning	% Response	Response File Size	Minutes to Get Response to Repository	Type of Response	Response Priority	
1	Snapshot400	CANCOM	90	100	60	Imagery	400	
2	Snapshot400	CEFCOM	90	100	80	Imagery	400	
3	Snapshot400	JTFA	90	100	30	Imagery	400	
4	Snapshot400	JTFP	90	100	30	Imagery	400	
5	Snapshot400	PRT	90	100	60	Imagery	400	
6	Snapshot400	Strategic	90	100	10	Imagery	400	
7	Imagery200	JTFP	90	100	60	Imagery	200	
8	Imagery200	JTFA	90	100	80	Imagery	200	
9	Imagery100	JTFP	90	100	30	Imagery	100	
10	Imagery100	JTFA	90	100	30	Imagery	100	
11	SITREP300	CANCOM	90	100	60	Imagery	300	
12	SITREP300	CEFCOM	90	100	10	Imagery	300	
13	SITREP300	JTFA	90	100	60	Imagery	300	
14	SITREP300	JTFP	90	100	80	Imagery	300	

Figure 31: R7-ExternalResponse Example

A.2.4 Execution

This section describes the process to run the SMOFN simulation. It assumes that the user has some knowledge in CORE and has the SMOFN loaded.

- Open the SMOFN project
- In the Project pane, select the class OperationalActivity
- In the Project pane, select the folder State Machine Sim
- In the Elements pane, select SM State Machine; the active window should look like Figure 32

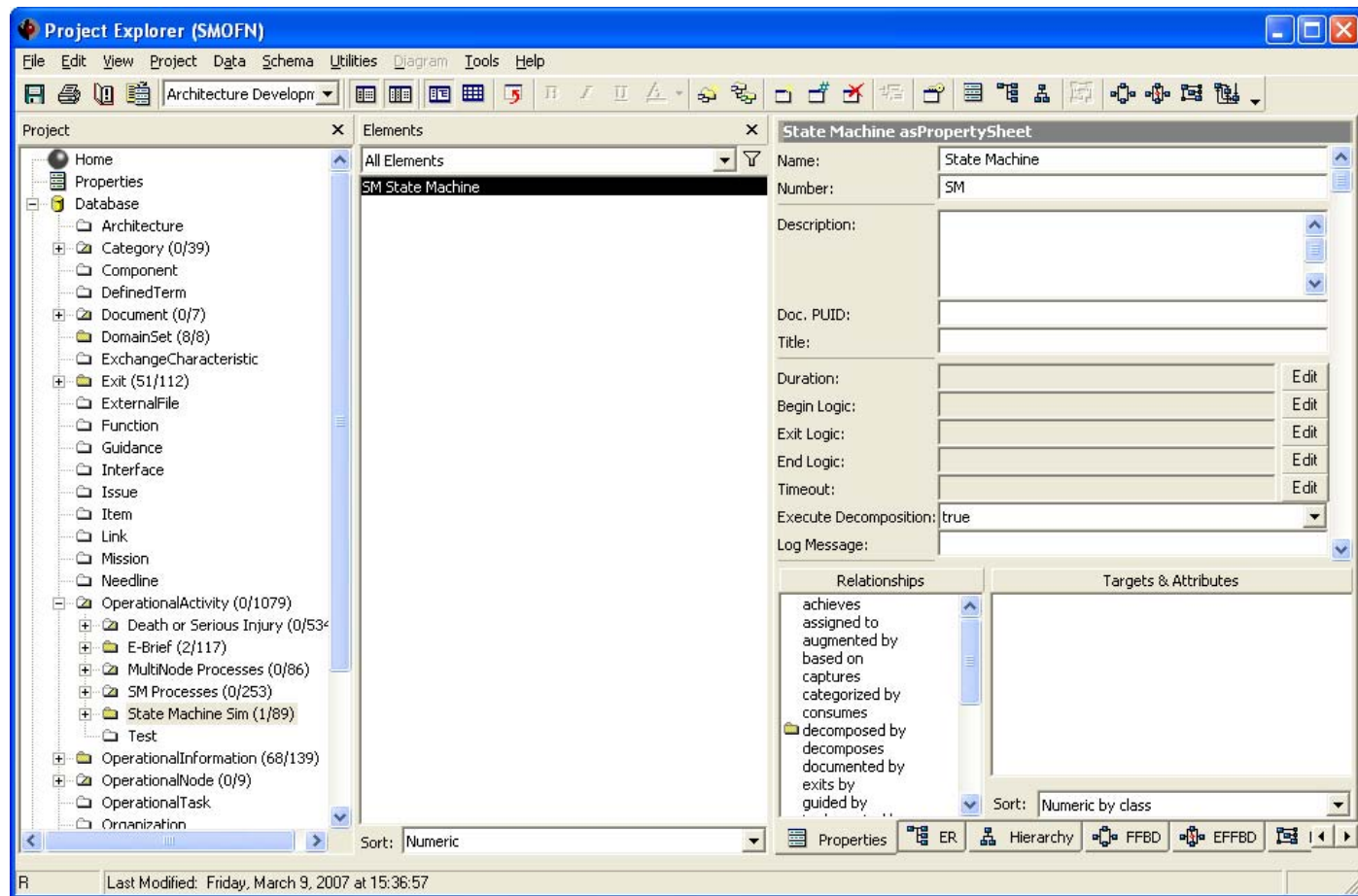


Figure 32: State Machine Element in CORE

- Under the Tools menu item, select Simulator
 - ♦ The Simulator Control Panel opens; it should look like Figure 33:

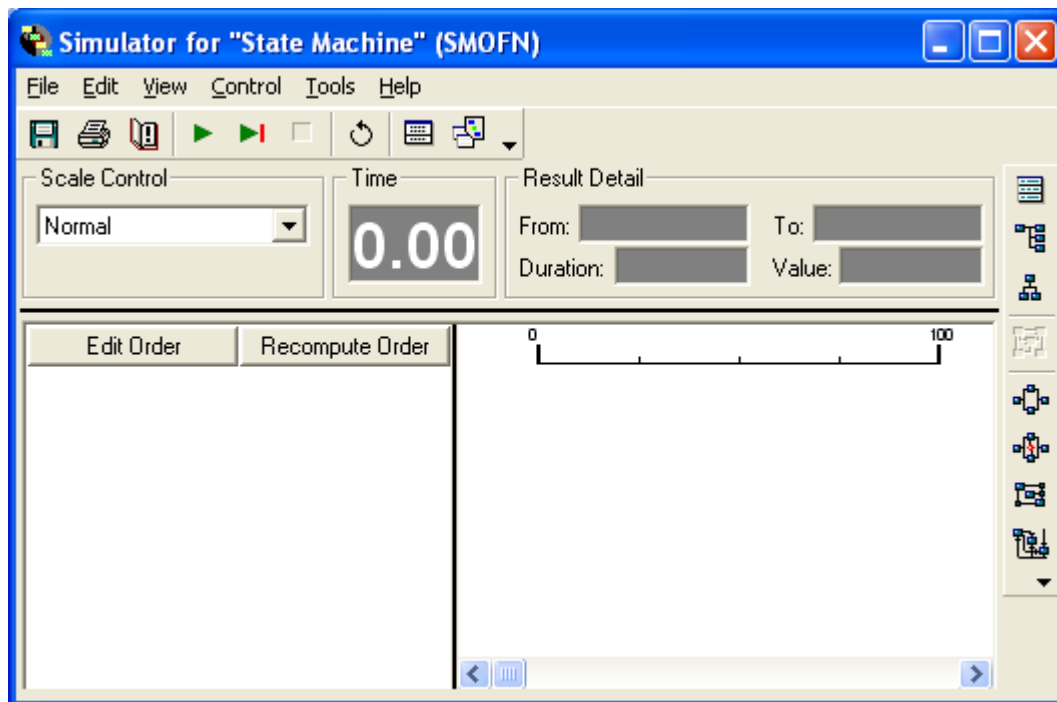


Figure 33: Simulator Control Panel

NOTE: It is strongly recommended that the simulator output be turned off for long durations of execution. Do this by selecting (on the Simulator Control Panel) View, Show Timeline Elements, and in the resulting Timeline Elements dialog box, click on Uncheck All in the Functions and Items panes.

- On the Simulator Control Panel, select Control, Run
 - ◆ Execution begins and the Open Data Path dialog box appears.
 - ◆ Navigate to the appropriate folder, select the file S1-Datapath.txt, and click the Open button
 - ◆ A CORESim Script Transcript window, shown in Figure 34, opens and initialization results are displayed, followed by execution processing of the scenario.

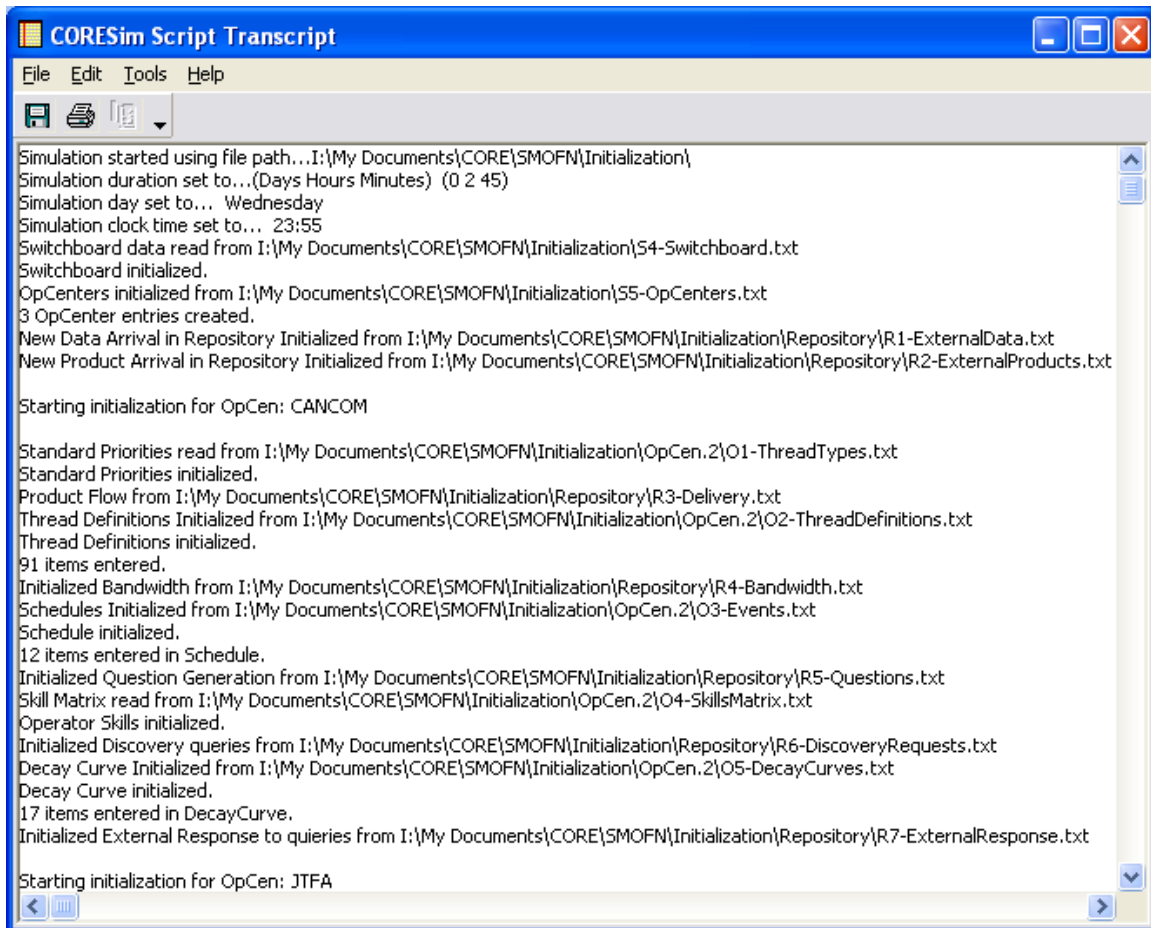


Figure 34: Simulation Output Transcript

- ◆ Depending on the length of the scenario, SimDuration (simulated execution time), and the numbers of events to be processed, the simulation runs for seconds to minutes

- ♦ The simulation runs until the current simulation time equals SimDuration, or until there are no events left to process, and it completes with the message shown in Figure 35.



Figure 35: Simulation Complete Dialog

- ♦ Click OK
- ♦ The dialog box Model Changes Detected, shown in Figure 36, may appear

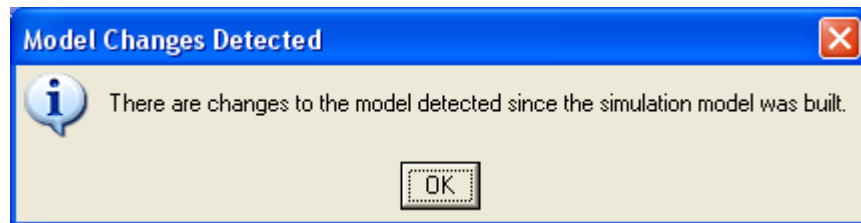


Figure 36: Model Changes Dialog Box

- ♦ Click OK

Suggestion: Save the Transcript as a text file at this point so that all events are easy to correlate later. It provides a good summary of event times that can be used to process the more detailed and complex Results files.

At this point the scenario simulation is completed and the Simulator Control Panel can be closed or another scenario executed.

A.2.5 Results

Seven results files are created in CSV format and would normally open in Excel.

Results1-Init.csv, shown in Figure 37, echoes the initialization data at the beginning of the transcript. It can be used for confirmation & troubleshooting if unexpected results are found.

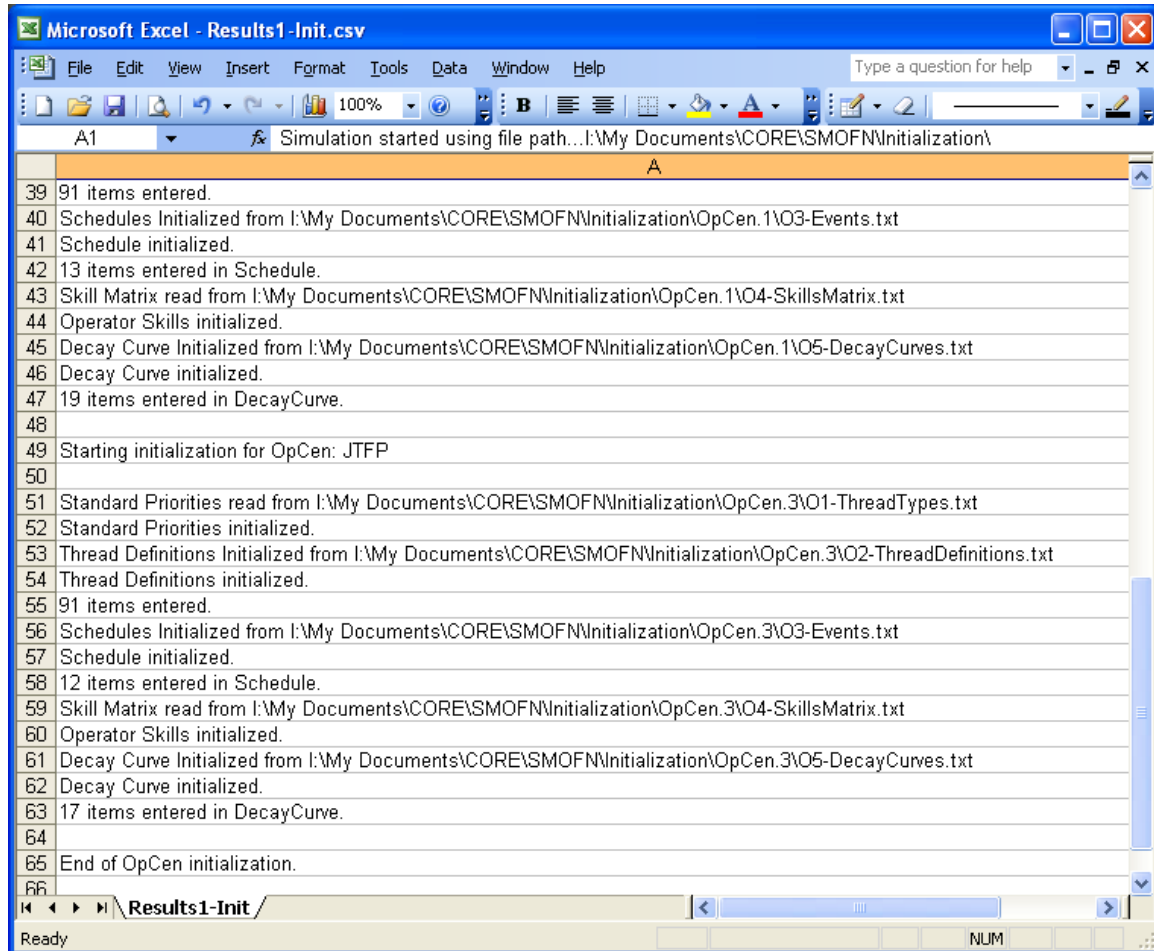
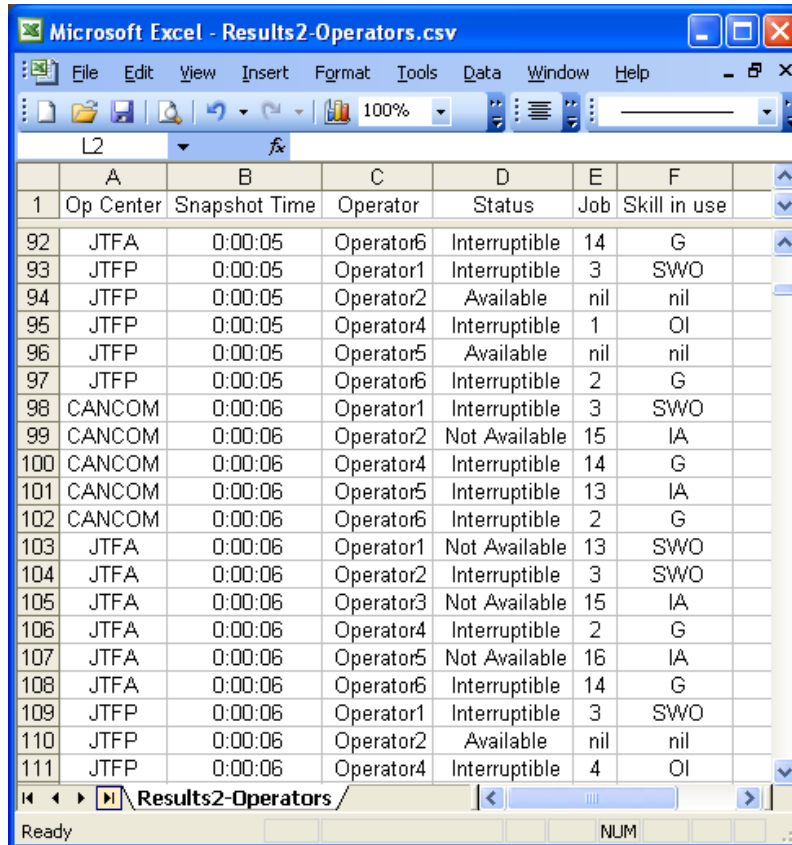


Figure 37: Results1-Init.csv Example

Results2-Operators.csv, shown in Figure 38, contains the current status of each operator at each snapshot interval. The first line contains column headers. Each subsequent entry contains: OPCEN, current simulation time, operator, operator status, current tasking (job number), and the skill that the operator is currently using.



	A	B	C	D	E	F
1	Op Center	Snapshot Time	Operator	Status	Job	Skill in use
92	JTFA	0:00:05	Operator6	Interruptible	14	G
93	JTFP	0:00:05	Operator1	Interruptible	3	SWO
94	JTFP	0:00:05	Operator2	Available	nil	nil
95	JTFP	0:00:05	Operator4	Interruptible	1	OI
96	JTFP	0:00:05	Operator5	Available	nil	nil
97	JTFP	0:00:05	Operator6	Interruptible	2	G
98	CANCOM	0:00:06	Operator1	Interruptible	3	SWO
99	CANCOM	0:00:06	Operator2	Not Available	15	IA
100	CANCOM	0:00:06	Operator4	Interruptible	14	G
101	CANCOM	0:00:06	Operator5	Interruptible	13	IA
102	CANCOM	0:00:06	Operator6	Interruptible	2	G
103	JTFA	0:00:06	Operator1	Not Available	13	SWO
104	JTFA	0:00:06	Operator2	Interruptible	3	SWO
105	JTFA	0:00:06	Operator3	Not Available	15	IA
106	JTFA	0:00:06	Operator4	Interruptible	2	G
107	JTFA	0:00:06	Operator5	Not Available	16	IA
108	JTFA	0:00:06	Operator6	Interruptible	14	G
109	JTFP	0:00:06	Operator1	Interruptible	3	SWO
110	JTFP	0:00:06	Operator2	Available	nil	nil
111	JTFP	0:00:06	Operator4	Interruptible	4	OI

Figure 38: Results2-Operators.csv Example

Results3-Events.csv, shown in Figure 39, contains the current status of each event at each snapshot interval. The first line contains column headers. Each subsequent entry has the detail: OPCEN, Job Number, Current Time, Event Type, Event Priority, Current Status, Accrued Utility, Decayed Utility, Net Utility, Current Operator, Number of Operators, Number of Times Interrupted, Estimated Time of Completion, Drop Dead Time, Slack Time, Time Spent in current step or queue, Completion Time, and Step Times for each step in the job thread. Each Step Time has three entries: step name, step start time and step completion time.

1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	Op Center	Job #	Clock Time	Event Type	Event Priority	Current Status	Accrued Utility	Decayed Utility	Net Utility	Operator (Current)	Operators (Number)	Times Interrupted	ETC	Drop Dead Time	Slack Time	Time Spent	Completed @ Step Times
949	JTFA	24	0:00:52	Imagery100	100	COMPLETE	1.4395	0.01	0.014		3	0	0:00:52.0	0:0:57	5		0:00:52 Complete: 0:00:52
950	JTFP	0	0:00:52														
951	CANCOM	13	0:00:53	Imagery200	200	Step_8	0.87	0.96	0.835	Operator5	3	2	0:01:05.0	0:01:19 (calculated)	14	3	Step: START Start:
952	CANCOM	20	0:00:53	Imagery200	200	Step_5	0.567	0.77	0.437	Operator4	3	1	0:01:21.0	0:01:33 (calculated)	12	2	Step: START Start:
953	CANCOM	24	0:00:53	Discovery100	100	Provide_Directic	0	1	0	Operator1	1	0	0:01:17.0	0:01:47 (calculated)	30	4	Step: CM_Review S
954	JTFA	11	0:00:53	SITREP100	100	Step_6	11.632	1	11.63	Operator1	3	0	0:00:57.0	0:00:57	6	1	Step: START Start:
955	JTFA	13	0:00:53	Discovery100	100	ulate_New_De	0	1	0	Operator4	3	0	0:01:01.0	0:01:02 (calculated)	19	19	Step: CM_Review S
956	JTFA	14	0:00:53	Imagery200	200	Step_7	0.912	0.97	0.885	Operator2	3	1	0:01:11.0	0:01:19 (calculated)	8	2	Step: START Start:
957	JTFA	16	0:00:53	Imagery200	198	COMPLETE	0.9465	0.83	0.786		4	0	0:00:59.0	0:01:23 (calculated)	24		0:00:53 Complete: 0:00:53
958	JTFP	0	0:00:53														
959	CANCOM	13	0:00:54	Imagery200	200	Step_9	0.942	0.96	0.904	Operator5	3	2	0:01:03.0	0:01:19 (calculated)	16	1	Step: START Start:
960	CANCOM	20	0:00:54	Imagery200	200	Step_5	0.567	0.77	0.437	Operator4	3	1	0:01:21.0	0:01:33 (calculated)	12	3	Step: START Start:
961	CANCOM	24	0:00:54	Discovery100	100	Provide_Directic	0	1	0	Operator1	1	0	0:01:17.0	0:01:47 (calculated)	30	5	Step: CM_Review S
962	JTFA	11	0:00:54	SITREP100	100	Step_7	11.762	1	11.76	Operator1	3	0	0:00:57.0	0:00:57	6	1	Step: START Start:
963	JTFA	13	0:00:54	Discovery100	100	ulate_New_De	0	1	0	Operator4	3	0	0:01:01.0	0:01:02 (calculated)	19	20	Step: CM_Review S
964	JTFA	14	0:00:54	Imagery200	200	Step_7	0.912	0.97	0.885	Operator2	3	1	0:01:11.0	0:01:19 (calculated)	8	3	Step: START Start:
965	JTFP	0	0:00:54														
966	CANCOM	13	0:00:55	Imagery200	200	Step_9	0.942	0.96	0.904	Operator5	3	2	0:01:03.0	0:01:19 (calculated)	16	2	Step: START Start:
967	CANCOM	20	0:00:55	Imagery200	200	Step_5	0.567	0.77	0.437	Operator4	3	1	0:01:21.0	0:01:33 (calculated)	12	4	Step: START Start:

Figure 39: Results3-Events.csv example

Results4-Events.Med.csv, shown in Figure 40, is formatted identically to Results3-Events.csv, however, it contains only those lines corresponding to a change in a job's status.

1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	Op Center	Job #	Clock Time	Event Type	Event Priority	Current Status	Accrued Utility	Decayed Utility	Net Utility	Operator (Current)	Operators (Number)	Times Interrupted	ETC	Drop Dead Time	Slack Time	Time Spent	Completed @ Step Times
343	CANCOM	24	0:00:50	Discovery100	100	Provide_Directic	0	1	0	Operator1	1	0	0:01:17.0	0:01:47 (calculated)	30	1	Step: CM_Review
344	JTFA	24	0:00:50	Imagery100	100	Step_8	1.2375	0.01	0.01	Operator2	2	0	0:00:53.0	0:0:57	4	4	Step: START Start:
345	CANCOM	13	0:00:51	Imagery200	200	Step_8	0.798	0.96	0.77	Operator5	3	2	0:01:05.0	0:01:19 (calculated)	14	1	Step: START Start:
346	JTFA	14	0:00:51	Imagery200	200	Step_6	0.622	0.97	0.6	Operator2	3	1	0:01:13.0	0:01:19 (calculated)	6	1	Step: START Start:
347	JTFA	24	0:00:51	Imagery100	100	Step_9	1.2775	0.01	0.01	Operator5	3	0	0:00:52.0	0:0:57	5	1	Step: START Start:
348	CANCOM	20	0:00:52	Imagery200	200	Step_5	0.567	0.77	0.44	Operator4	3	1	0:01:21.0	0:01:33 (calculated)	12	1	Step: START Start:
349	JTFA	11	0:00:52	SITREP100	100	Step_5	11.515	1	11.5	Operator5	3	0	0:00:57.0	0:00:57	6	1	Step: START Start:
350	JTFA	14	0:00:52	Imagery200	200	Step_7	0.762	0.97	0.74	Operator2	3	1	0:01:11.0	0:01:19 (calculated)	8	1	Step: START Start:
351	JTFA	24	0:00:52	Imagery100	100	COMPLETE	1.4395	0.01	0.01		3	0	0:00:52.0	0:0:57	5		0:00:52 Complete: 0:00:52
352	CANCOM	13	0:00:53	Imagery200	200	Step_8	0.87	0.96	0.84	Operator5	3	2	0:01:05.0	0:01:19 (calculated)	14	3	Step: START Start:
353	JTFA	11	0:00:53	SITREP100	100	Step_6	11.632	1	11.6	Operator1	3	0	0:00:57.0	0:00:57	6	1	Step: START Start:
354	JTFA	14	0:00:53	Imagery200	200	Step_7	0.912	0.97	0.88	Operator2	3	1	0:01:11.0	0:01:19 (calculated)	8	2	Step: START Start:
355	JTFA	16	0:00:53	Imagery200	198	COMPLETE	0.9465	0.83	0.79		4	0	0:00:59.0	0:01:23 (calculated)	24		0:00:53 Complete: 0:00:53
356	CANCOM	13	0:00:54	Imagery200	200	Step_9	0.942	0.96	0.9	Operator5	3	2	0:01:03.0	0:01:19 (calculated)	16	1	Step: START Start:
357	JTFA	11	0:00:54	SITREP100	100	Step_7	11.762	1	11.8	Operator1	3	0	0:00:57.0	0:00:57	6	1	Step: START Start:
358	CANCOM	24	0:00:55	Discovery100	100	CM_Review	0	1	0	Operator2	2	0	0:01:16.0	0:01:47 (calculated)	31	1	Step: CM_Review
359	JTFA	11	0:00:55	SITREP100	100	Step_8	11.852	1	11.9	Operator1	3	0	0:00:57.0	0:00:57	6	1	Step: START Start:
360	JTFA	14	0:00:55	Imagery200	200	Step_8	1.062	0.97	1.03	Operator2	3	1	0:01:09.0	0:01:19 (calculated)	10	1	Step: START Start:
361	CANCOM	13	0:00:56	Imagery200	200	Step_9	0.978	0.96	0.94	Operator5	3	2	0:01:03.0	0:01:19 (calculated)	16	3	Step: START Start:

Figure 40: Results4-Events.Med.csv Example

Results5-Events.Short.csv, shown in Figure 41, is formatted identically to Results3-Events.csv except that it does not identify the current operator or the amount of time spent in the current step or queue. It contains only the final status for each job.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Op Center	Job #	Clock Time	Event Type	Event Priority	Current Status	Accrued Utility	Decayed Utility	Net Utility	Operators (Number)	Times Interrupted	ETC	Drop Dead Time	Slack Time	Completed @	Step Times
47	JTFP	9	0:00:40	Assessment100	100	COMPLETE	4.5092	1	4.51	2	0	0:00:41.0	0:00:43 (calculated)	6.5	0:00:40	Complete: 0:00:40
48	CANCOM	9	0:00:42	Assessment100	100	COMPLETE	7.24	1	7.24	4	0	0:00:43.0	0:00:43 (calculated)	3	0:00:42	Complete: 0:00:42
49	JTFP	2	0:00:42	SITREP300	300	COMPLETE	1.3552	1	1.36	3	0	0:00:45.0	0:01:04 (calculated)	22	0:00:42	Complete: 0:00:42
50	CANCOM	23	0:00:43	Imagery100	100	COMPLETE	1.295	0.31	0.4	3	0	0:00:43.0	0:0:57	14	0:00:43	Complete: 0:00:43
51	JTFA	8	0:00:45	SITREP300	300	Abandoned	0.16	1	0.16	1	0	0:01:51.0	0:01:17 (calculated)	-28		Step: START Start: nil Finish: nil
52	JTFA	21	0:00:48	Imagery200	200	NSTR	0.9	0.93	0.84	1	0	0:01:31.0	0:01:33 (calculated)	2		Step: START Start: nil Finish: nil
53	JTFA	24	0:00:52	Imagery100	100	COMPLETE	1.4395	0.01	0.01	3	0	0:00:52.0	0:0:57	5	0:00:52	Complete: 0:00:52
54	JTFA	16	0:00:53	Imagery200	198	COMPLETE	0.9465	0.83	0.79	4	0	0:00:59.0	0:01:23 (calculated)	24	0:00:53	Complete: 0:00:53
55	CANCOM	13	0:00:57	Imagery200	200	COMPLETE	1.014	0.96	0.97	3	2	0:01:03.0	0:01:19 (calculated)	16	0:00:57	Complete: 0:00:57
56	JTFA	11	0:00:57	SITREP100	100	COMPLETE	12.115	1	12.1	3	0	0:00:57.0	0:00:57	6	0:00:57	Complete: 0:00:57
57	JTFA	13	0:01:01	Discovery100	100	COMPLETE	0	1	0	3	0	(0 1 1)	0:01:02 (calculated)	19	0:01:01	Complete: 0:01:01
58	JTFA	14	0:01:01	Imagery200	200	COMPLETE	1.309	0.38	0.5	3	1	0:01:07.0	0:01:19 (calculated)	12	0:01:01	Complete: 0:01:01
59	CANCOM	20	0:01:13	Imagery200	200	COMPLETE	1.502	0.18	0.27	3	1	0:01:19.0	0:01:33 (calculated)	14	0:01:13	Complete: 0:01:13
60	CANCOM	24	0:01:17	Discovery100	100	COMPLETE	0	1	0	3	0	(0 1 17)	0:01:47 (calculated)	48	0:01:17	Complete: 0:01:17
61	JTFA	25	0:01:34	Imagery100	100	NSTR	0.5	1	0.5	1	0	0:02:03.0	0:02:25 (calculated)	22		Step: START Start: nil Finish: nil
62	JTFA	6	0:01:40	SITREP100	100	COMPLETE	3.7035	0.03	0.11	3	0	0:01:40.0	0:01:51 (calculated)	15	0:01:40	Complete: 0:01:40
63	CANCOM	25	0:01:44	Discovery100	100	COMPLETE	0	1	0	2	0	(0 1 44)	0:02:14 (calculated)	48	0:01:44	Complete: 0:01:44
64																
65																

Figure 41: Results5-Events.Short.csv Example

Results6-Full.Aggregation.csv, shown in Figure 42, contains details of the aggregation of completed products into reports. The first line contains column headers. Each subsequent entry contains: OPCEN, Aggregated Job Number, Aggregating Job Number, Number of Posts made to the Repository, Number of Repository Posts that were expected, the Accrued Utility that was transferred to the Aggregating Job, and the Net Utility that was transferred.

	A	B	C	D	E	F	G	H
1	OPCEN	Aggregated Job	Aggregating Job	Posts Made	Posts Expected	Accrued Utility	Net Utility	
18	CANCOM	17	none					
19	CANCOM	18	9	1	1	0.5	0.5	
20	CANCOM	19	2	0	2	0.9	0.9	
21	CANCOM	20	none					
22	CANCOM	21	none					
23	CANCOM	22	none					
24	CANCOM	23	none					
25	CANCOM	24	none					
26	CANCOM	25	none					
27	JTFA	1	5	0	2	0.1	0.056	
28	JTFA	1	2	0	2	0.1	0.056	
29	JTFA	1	9	1	1	0.1	0.056	
30	JTFA	2	9	1	1	0.5	0.5	
31	JTFA	3	none					
32	JTFA	4	none					
33	JTFA	5	9	1	1	0.5	0.5	
34	JTFA	6	none					
35	JTFA	7	none					
36	JTFA	8	none					
37	JTFA	9	none					
38	JTFA	10	none					
39	JTFA	11	none					
40	JTFA	12	none					
41	JTFA	13	none					
42	JTFA	14	none					
43	JTFA	15	6	2	2	1.344	0.02688	

Figure 42: Results6-Full.Aggregation.csv Example

Results7-Partial.Aggregation.csv, shown in Figure 43, contains the same information as Results6-Full.Aggregation except that it lists the details of partially completed jobs that are aggregated, rather than finalized ones.

	A	B	C	D	E	F	G
		Aggregated Job	Aggregating Job	Posts Made	Posts Expected	Accrued Utility	Net Utility
1	OPCEN						
20	CANCOM	17	none				
21	CANCOM	18	none				
22	CANCOM	19	none				
23	CANCOM	20	2	0	2	0.207	0.207
24	CANCOM	21	none				
25	CANCOM	22	none				
26	CANCOM	23	none				
27	CANCOM	24	none				
28	CANCOM	25	none				
29	JTFA	1	none				
30	JTFA	2	none				
31	JTFA	3	none				
32	JTFA	4	none				
33	JTFA	5	none				
34	JTFA	6	none				
35	JTFA	7	none				
36	JTFA	8	none				
37	JTFA	9	none				
38	JTFA	10	none				
39	JTFA	11	none				
40	JTFA	12	none				
41	JTFA	13	none				
42	JTFA	14	2	0	2	0.28	0.28
43	JTFA	14	8	0	2	0.514	0.49858
44	JTFA	15	9	1	1	0.46	0.1242
45	JTFA	15	11	2	2	0.568	0.15336

Figure 43: Results7-Partial.Aggregation.csv Example

Annex B SMOFN Diagrams

This annex presents the Enhanced Functional Flow Block Diagrams (EFFBDs) used to control the SMOFN simulation sequence at all levels. Except for the Send and Receive activities for the Consumer and External Sources, whose logic is handled by the corresponding Repository activities (as noted in Section 3.2.5), all activities have either lower levels of decomposition or embedded code. While the decomposition of any operational activity is seen in the subsequent diagrams, the embedded code can be provided electronically.

Note that Figure 44 is the top level diagram as presented in Figure 5 of Chapter 3, however it is reproduced here without the annotation of the individual phases. Figure 45 through Figure 48 illustrate larger scale views of those phases. Figure 49 through Figure 55 illustrate lower levels of decomposition where any exists.

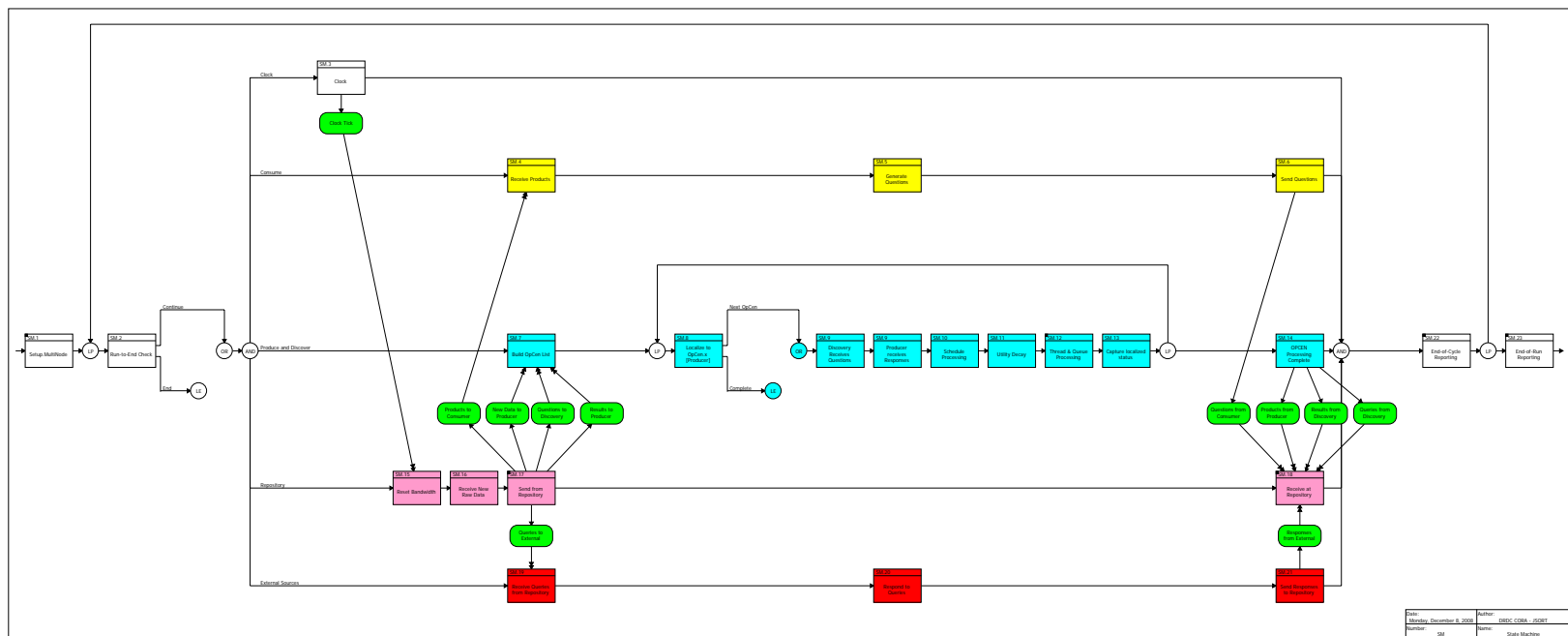


Figure 44: SM State Machine EFFBD

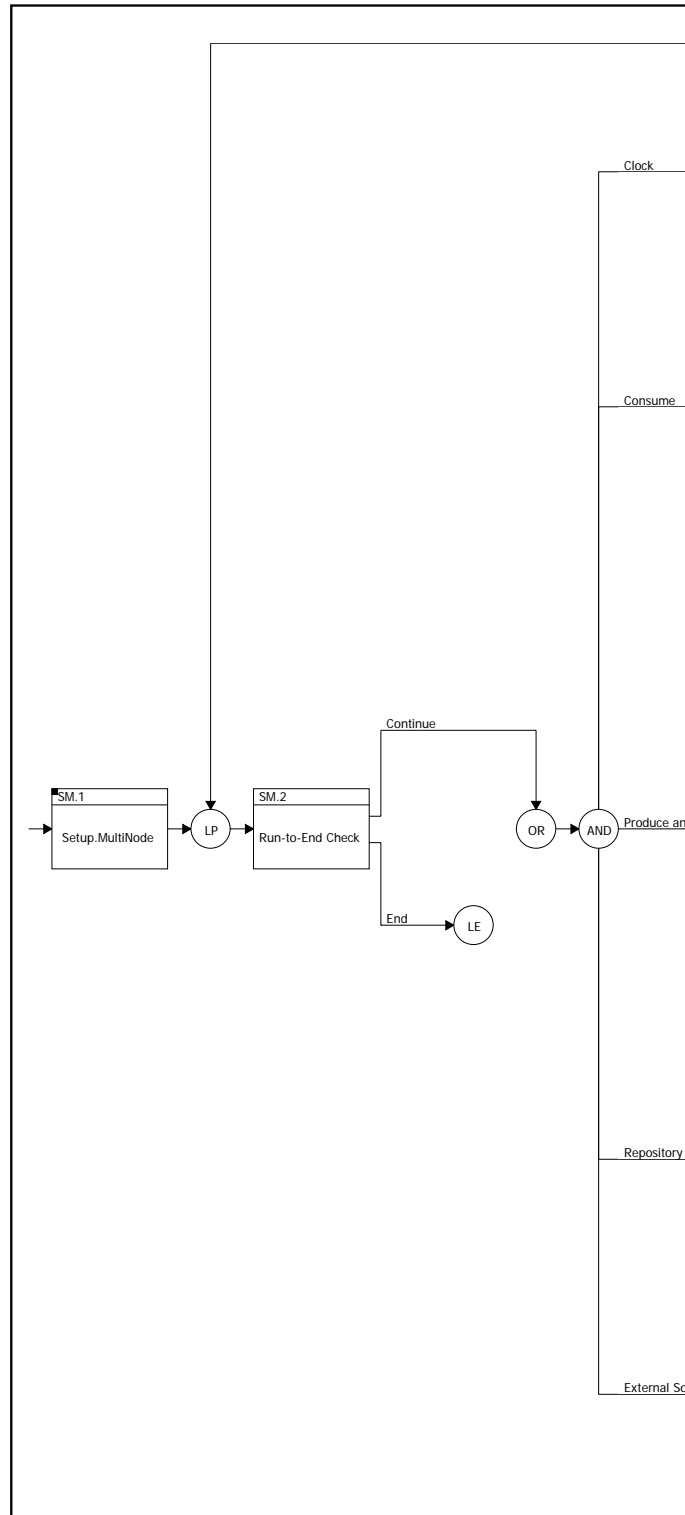


Figure 45: Simulation Control

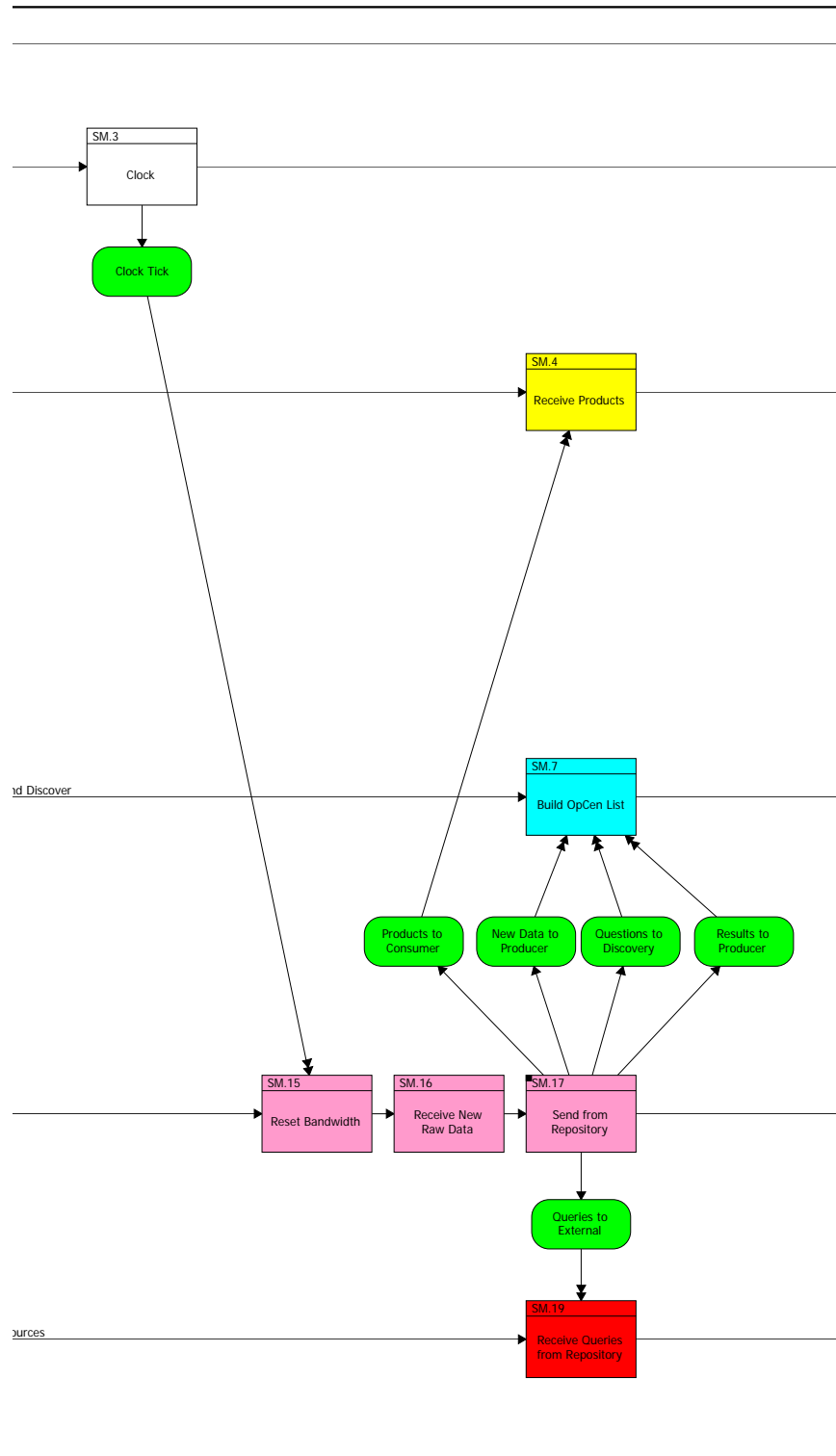


Figure 46: OPCEN Input

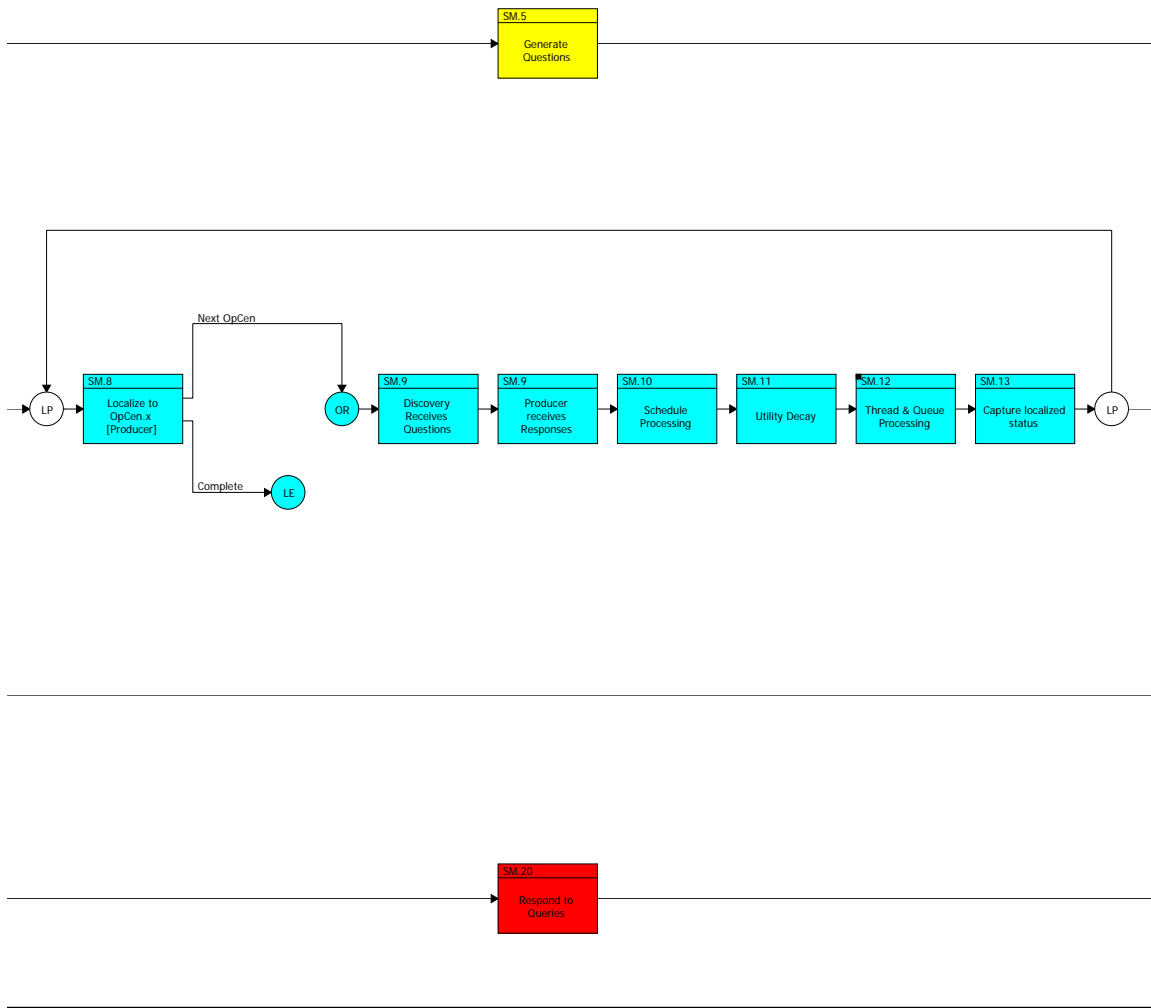


Figure 47: OPCEN Activity

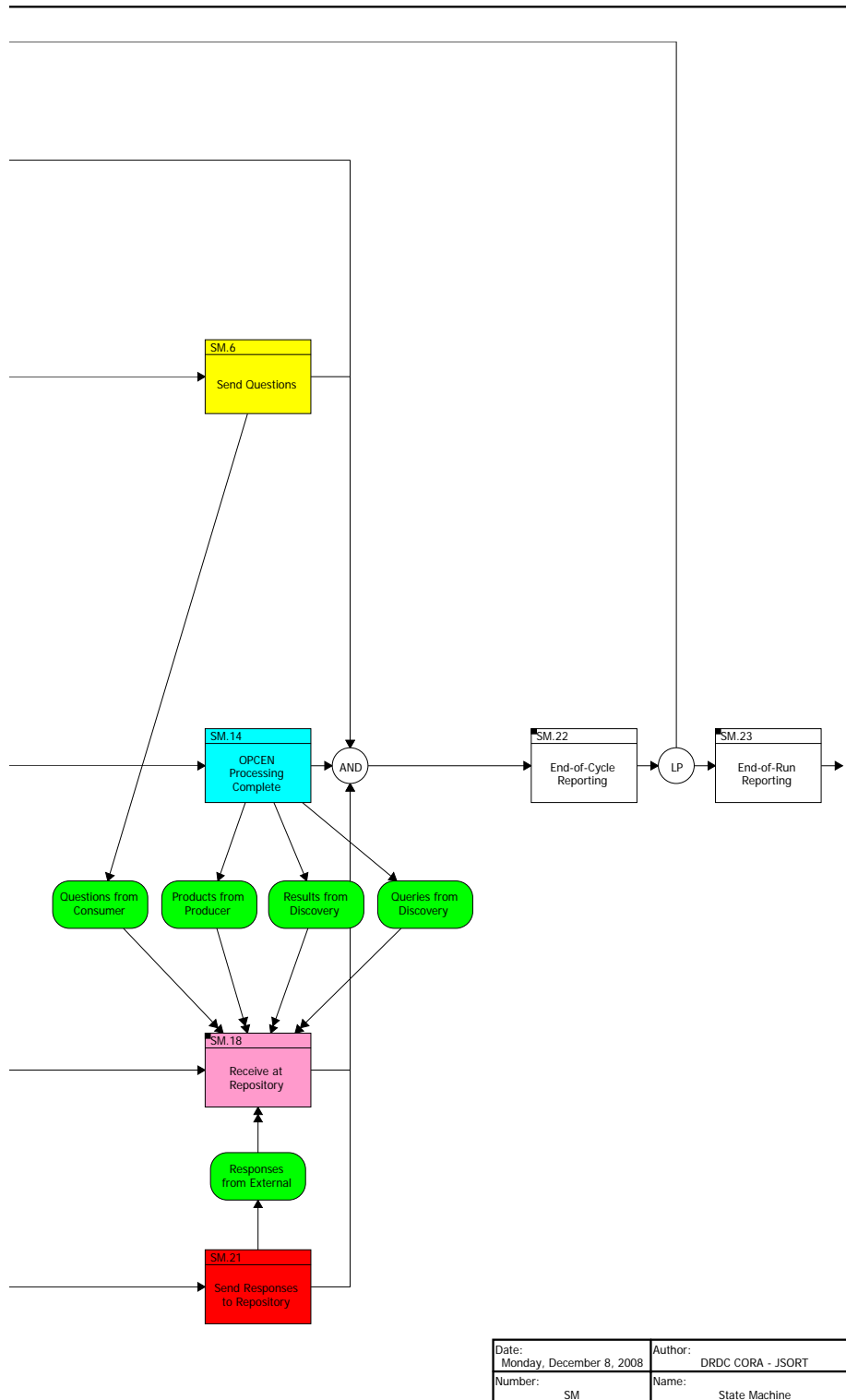


Figure 48: OPCEN Output and Simulation Output

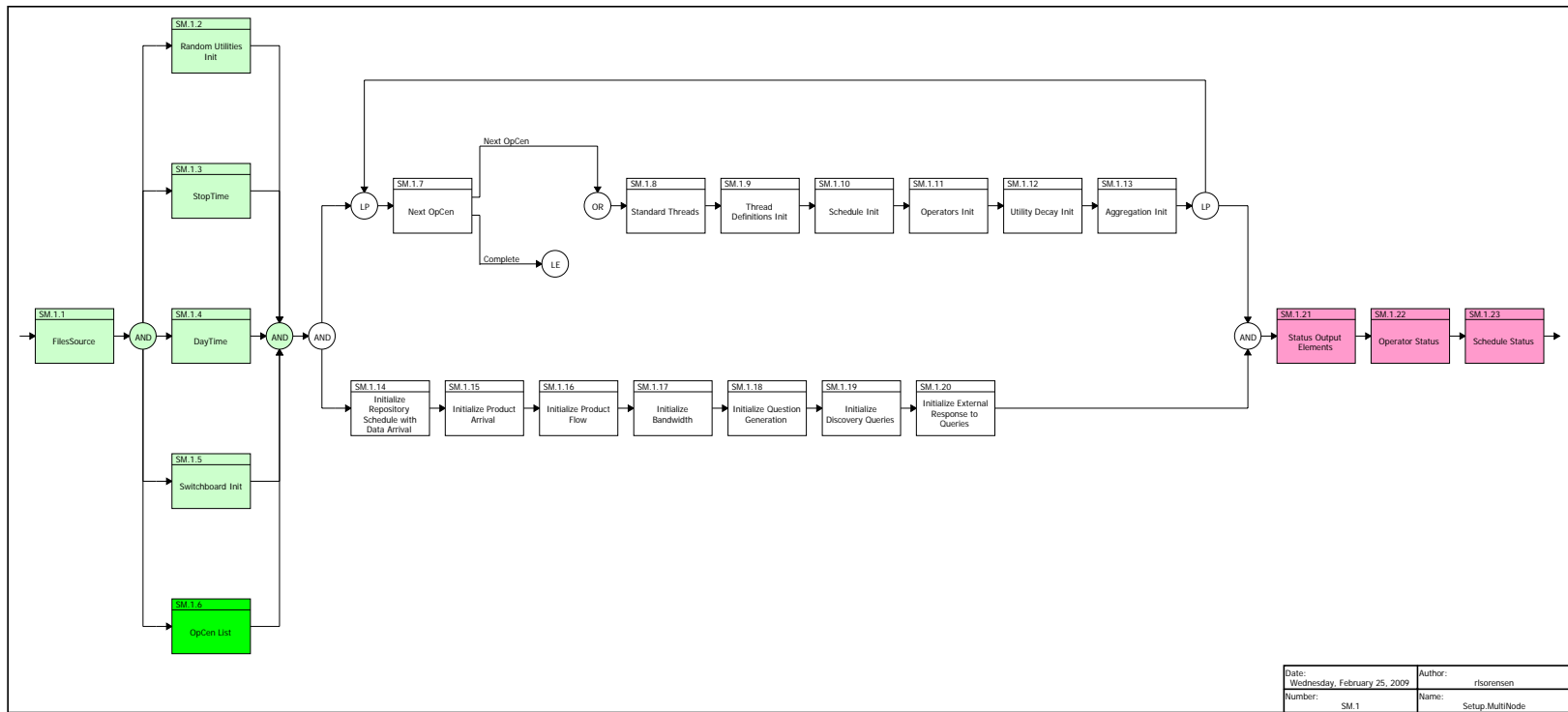


Figure 49: SM.1 Setup.MultiNode EFFBD

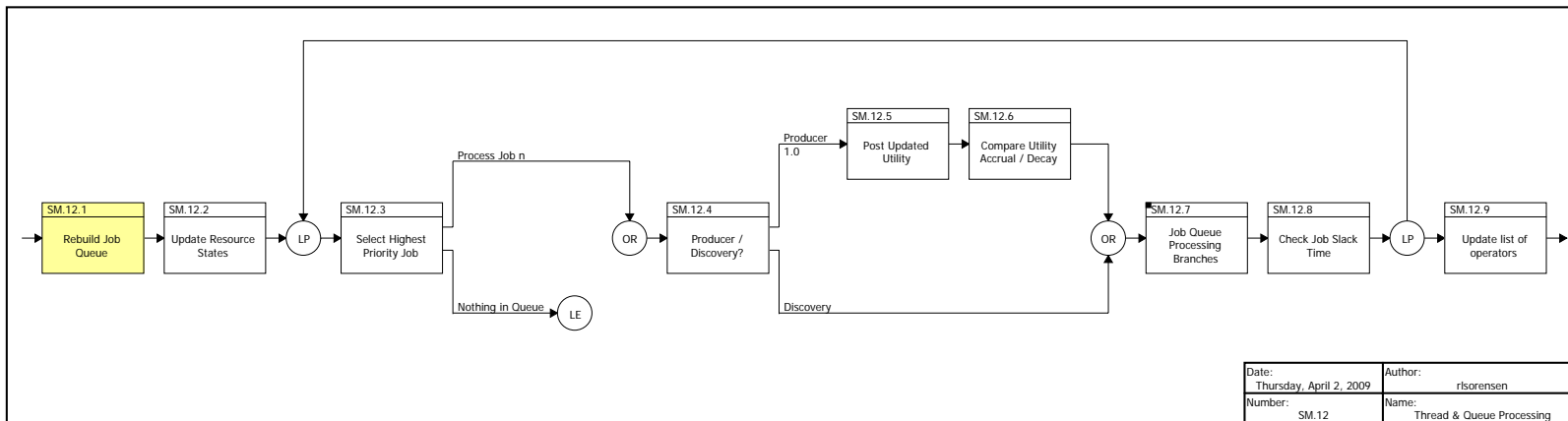


Figure 50: SM.12 Thread & Queue Processing EFFBD

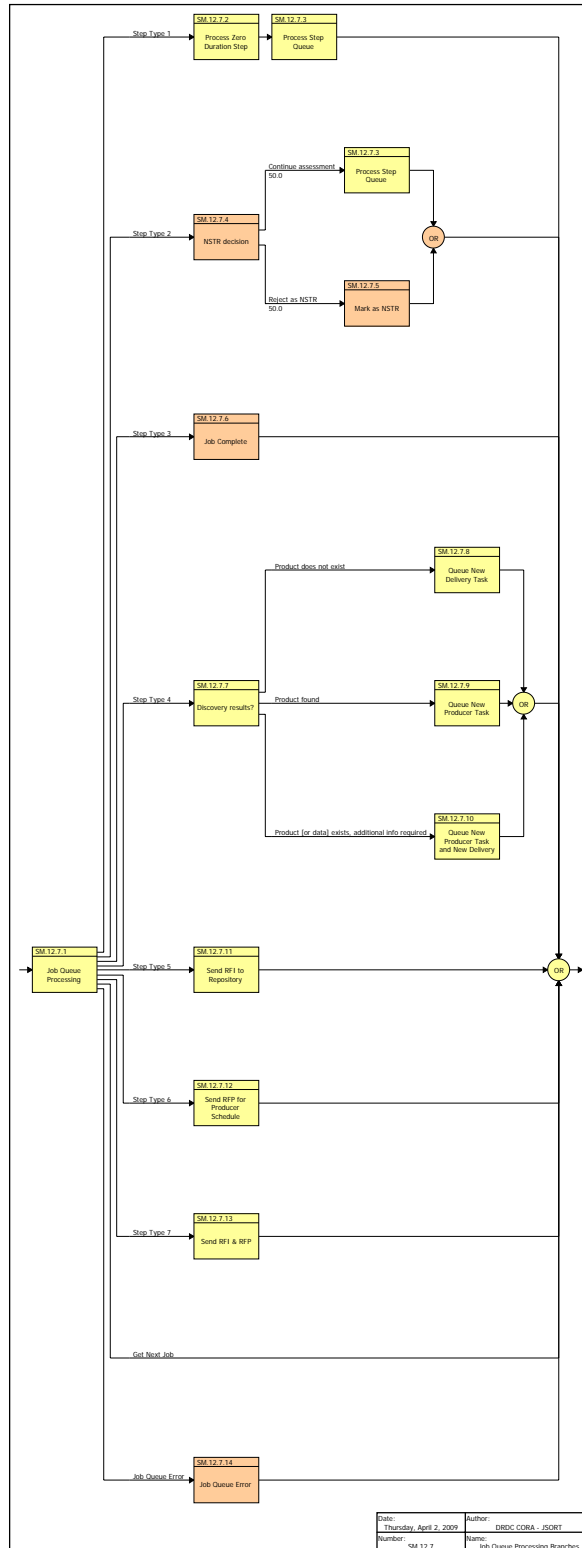


Figure 51: SM.12.7 Job Queue Processing Branches EFFBD

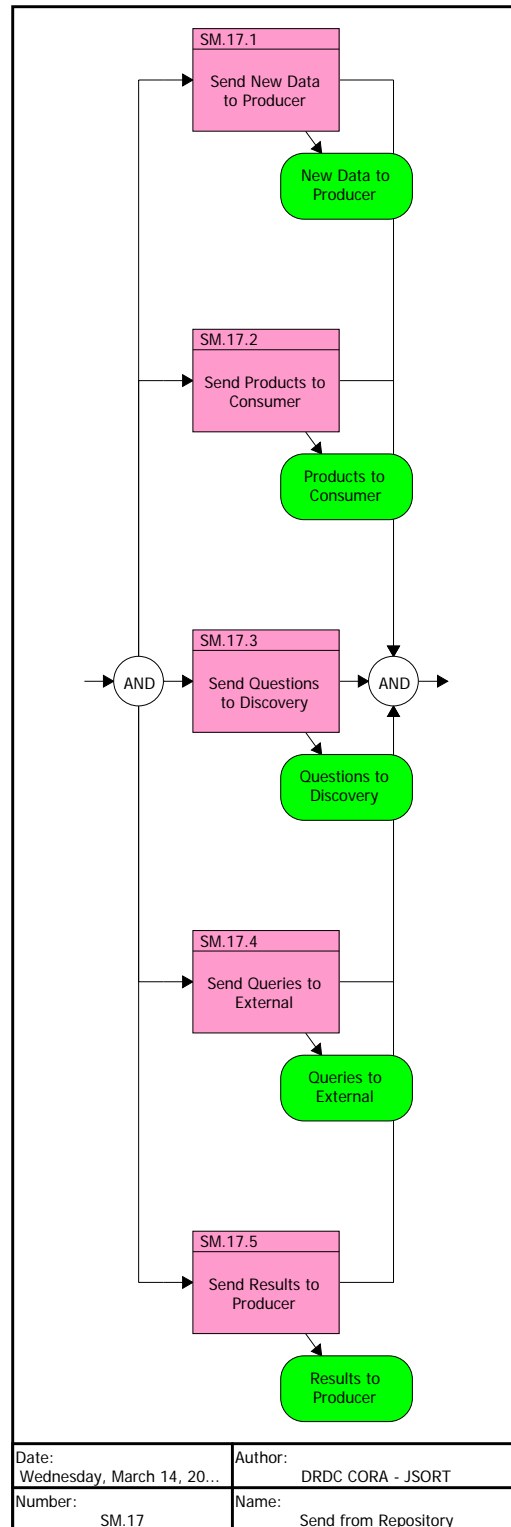


Figure 52: SM.17 Send from Repository EFFBD

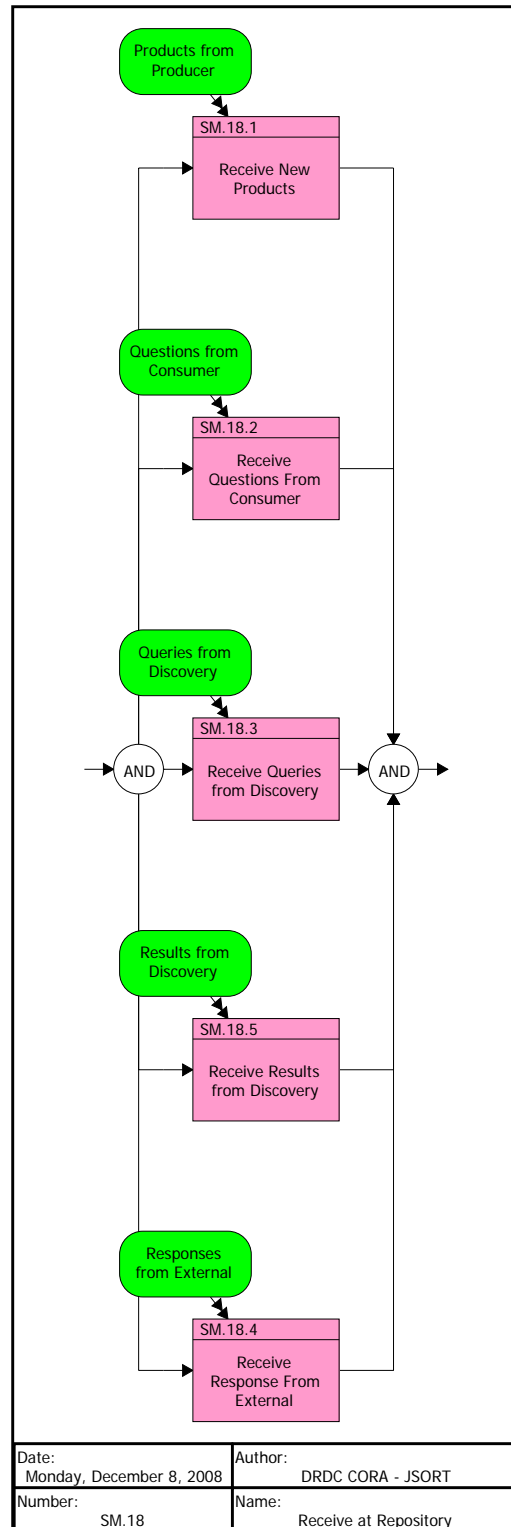


Figure 53: SM.18 Receive at Repository EFFBD

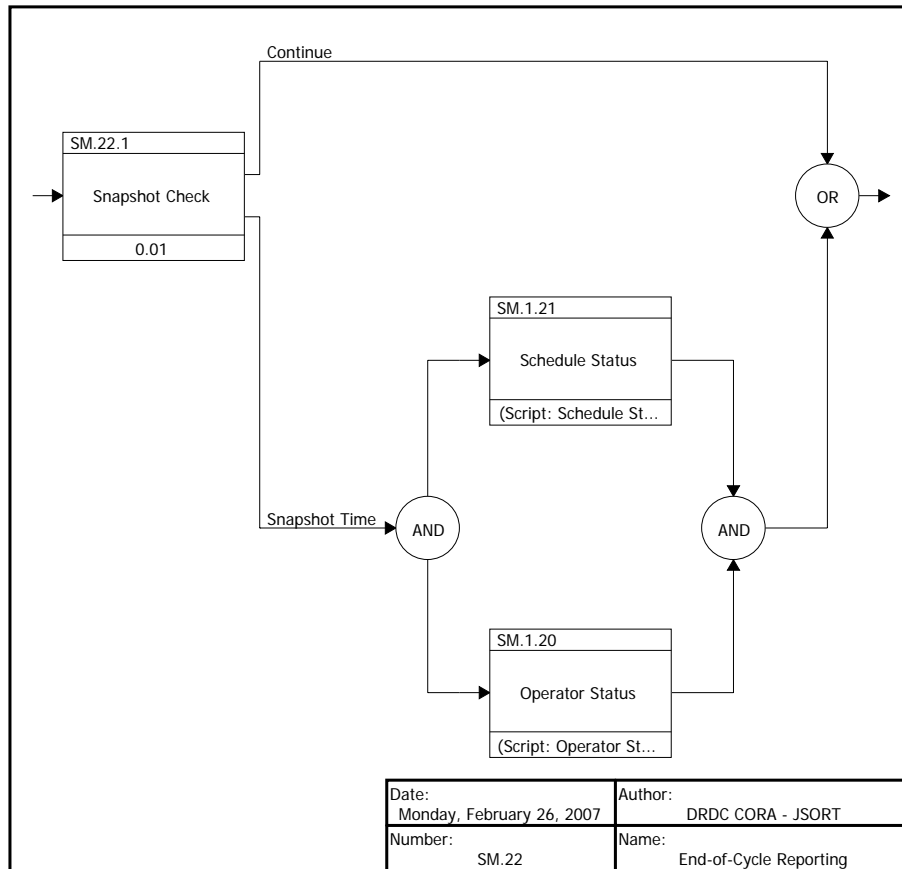


Figure 54: SM.22 End-of-Cycle Reporting EFFBD

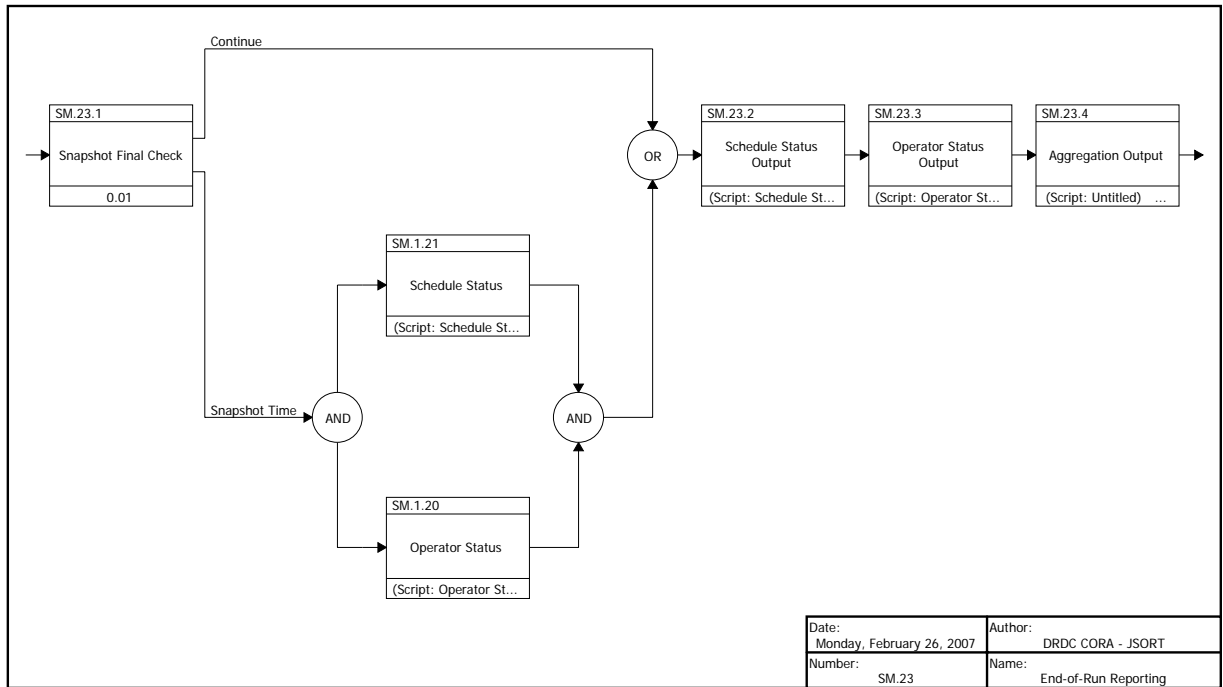


Figure 55: SM.23 End-of-Run Reporting EFFBD

Annex C Case 1 – eBrief

This Annex is included to provide all of the diagrams that make up the Electronic Brief preparation process model that was described in Section 6.1.2. Where decision logic is required (to choose the correct exit branch), the COREScript code is included.

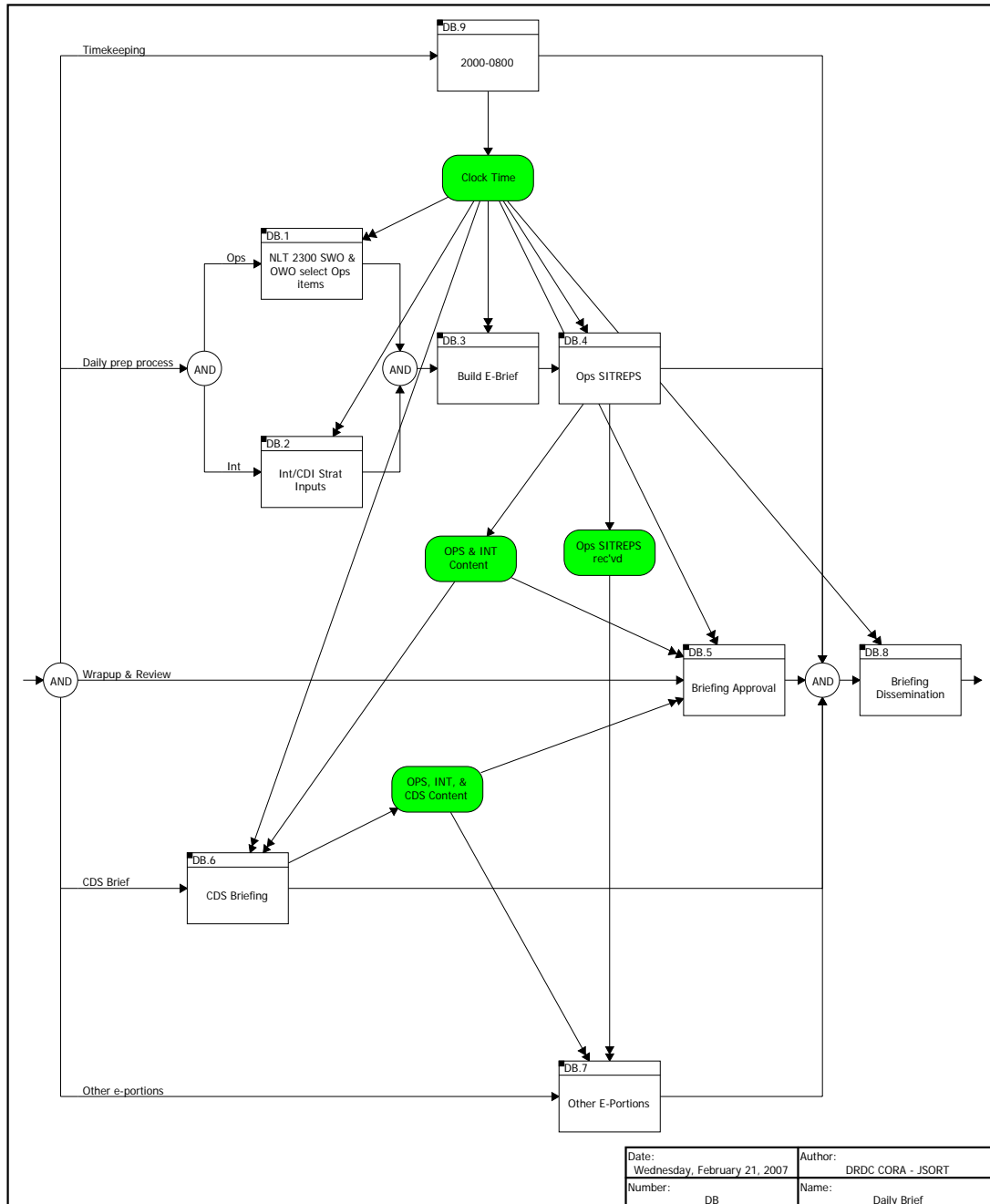


Figure 56: DB Daily Brief EFFBD

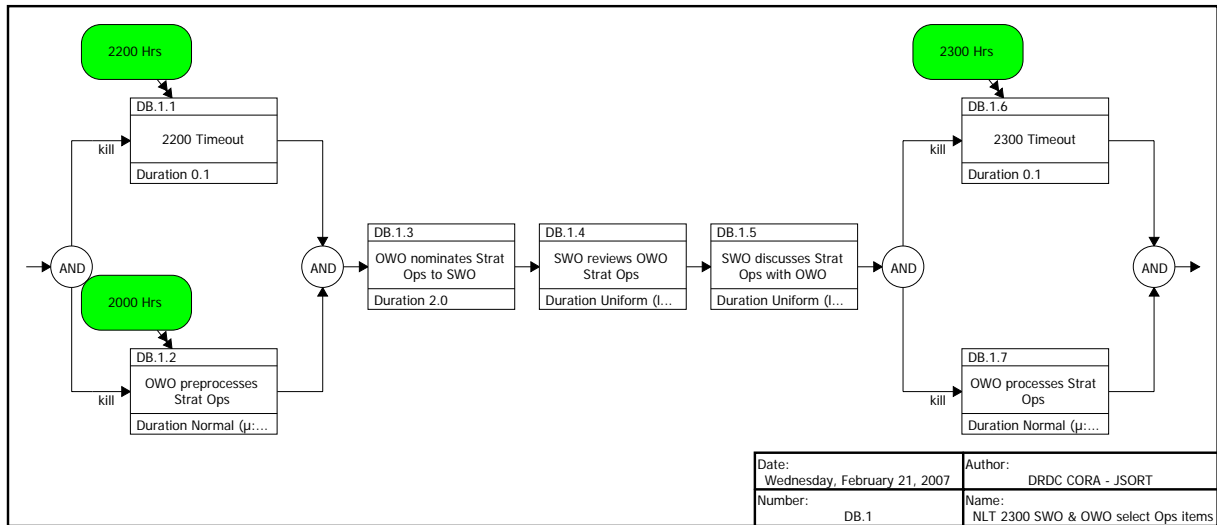


Figure 57: DB.1 NLT 2300 SWO & OWO select Ops items EFFBD

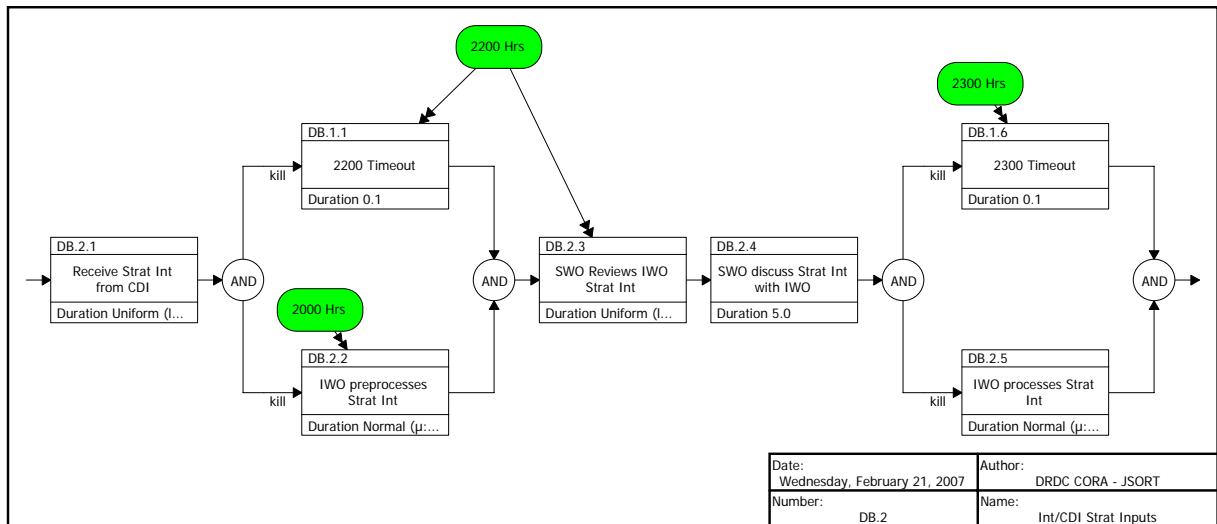


Figure 58: DB.2 Int/CDI Strat Inputs EFFBD

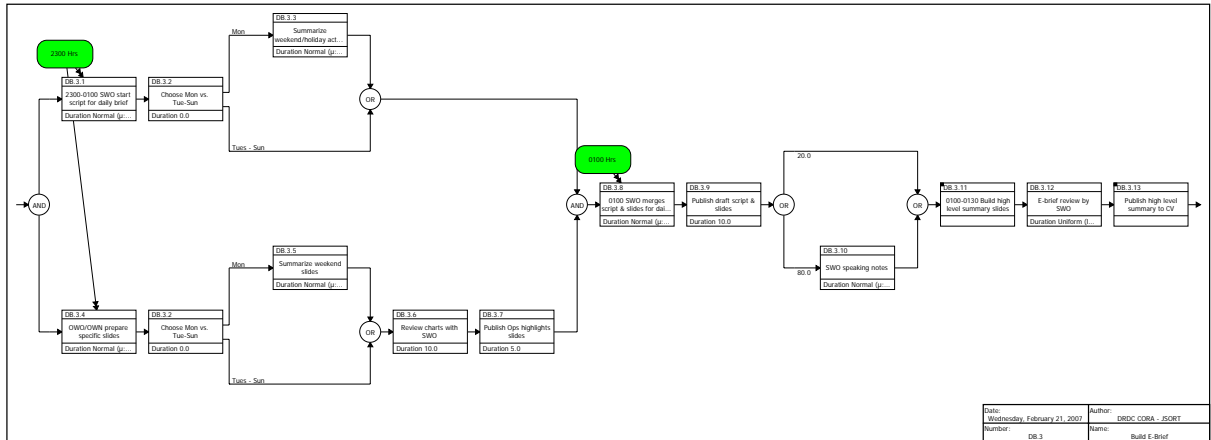


Figure 59: DB.3 Build E-Brief EFFBD

OperationalActivity Exit Logic Rules:

DB.3.2 Choose Mon vs. Tue-Sun

(Script: Untitled)

If (Day = 'Monday') Then

 exitLogic := NamedElement ("Exit", "Mon")

Else

 exitLogic := NamedElement ("Exit", "Tues - Sun")

EndIf

Return (exitLogic)

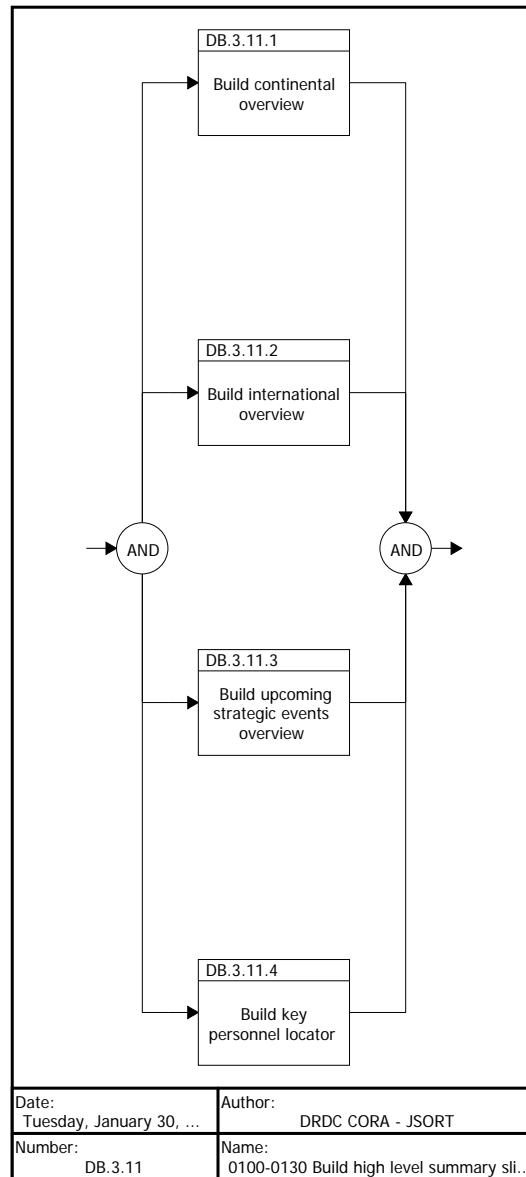


Figure 60: DB.3.11 0100-0130 Build high level summary slides EFFBD

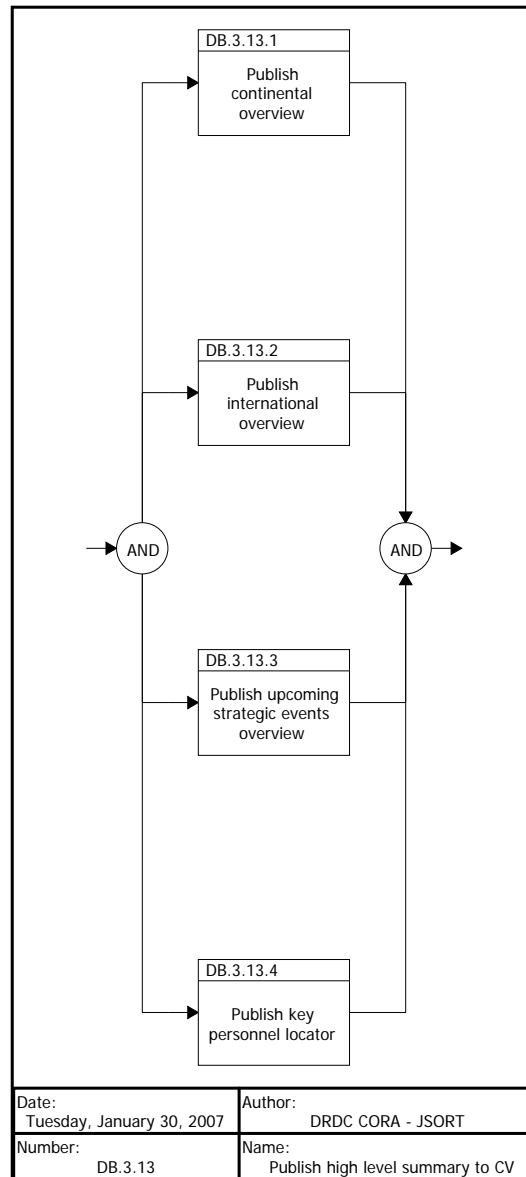


Figure 61: DB.3.13 Publish high level summary to CV EFFBD

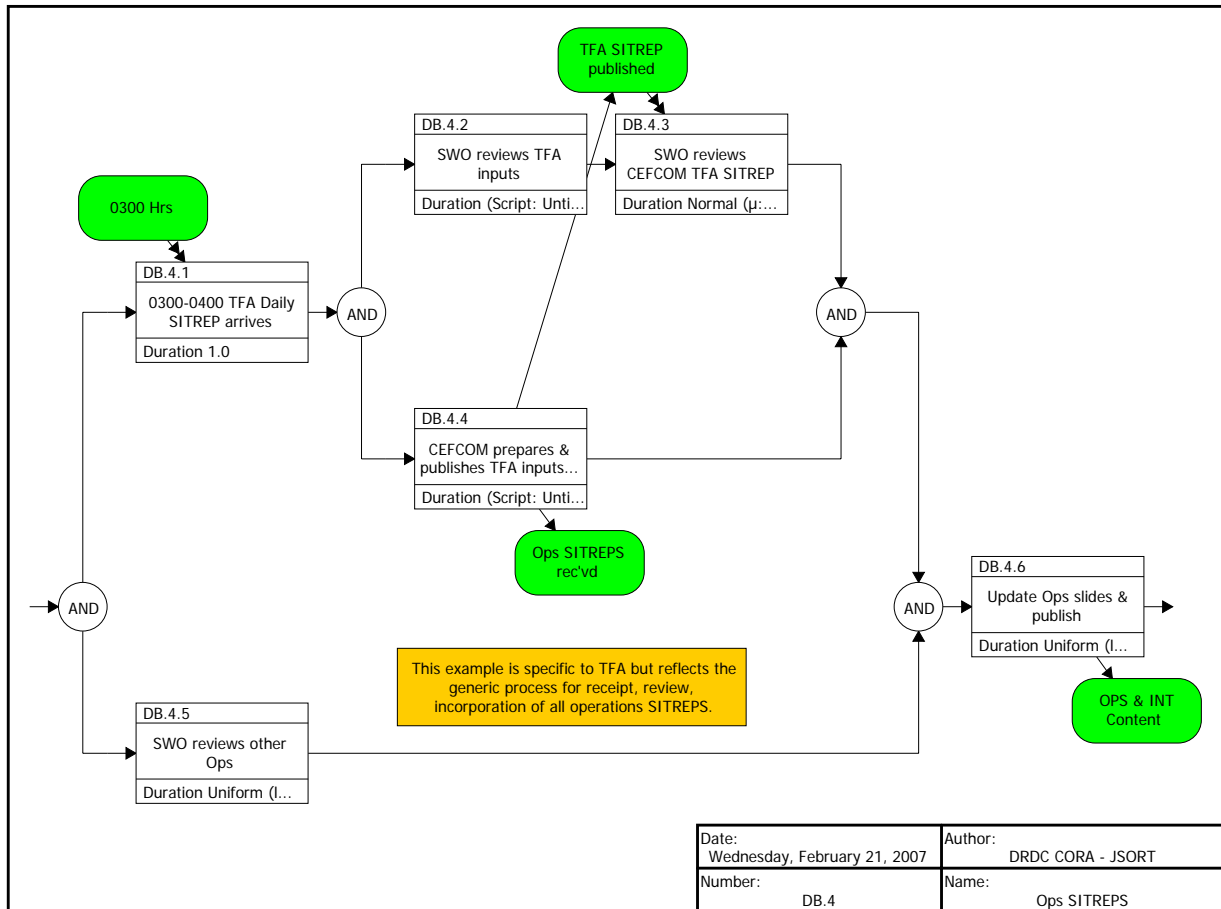


Figure 62: DB.4 Ops SITREPS EFFBD

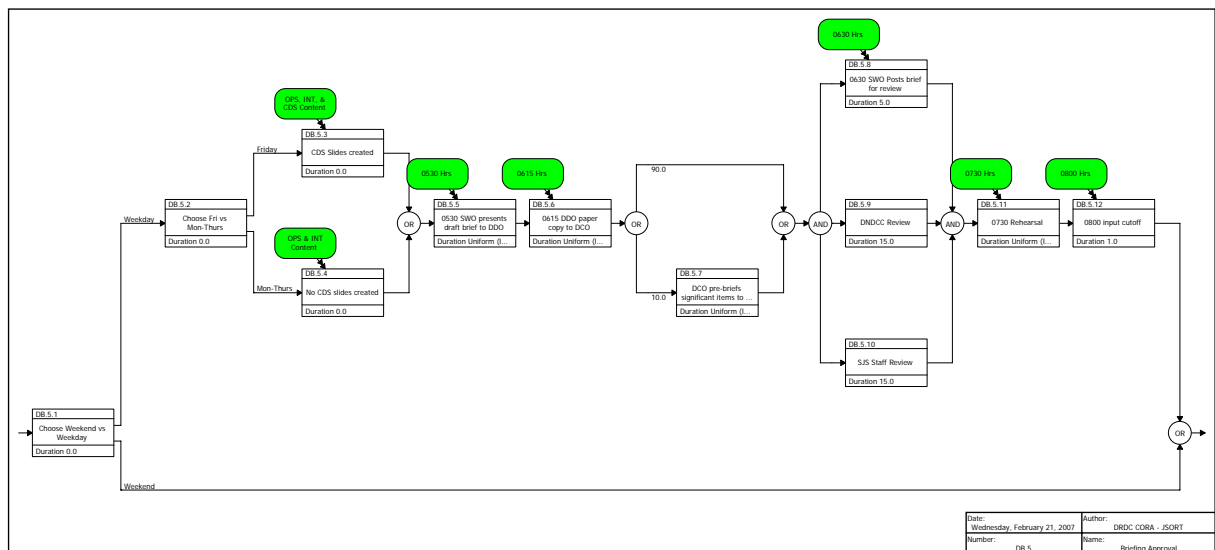


Figure 63: DB.5 Briefing Approval EFFBD

OperationalActivity Exit Logic Rules:

DB.5.1 Choose Weekend vs Weekday

```
(Script: Untitled)
If ((Day = 'Saturday') or: (Day = 'Sunday')) Then
    exitLogic := NamedElement ("Exit", "Weekend")
Else
    exitLogic := NamedElement ("Exit", "Weekday")
EndIf
Return (exitLogic)
```

DB.5.2 Choose Fri vs Mon-Thurs

```
(Script: Untitled)
If (Day = 'Friday') Then
    exitLogic := NamedElement ("Exit", "Friday")
Else
    If (((Day = 'Monday') or: (Day = 'Tuesday')) or: ((Day = 'Wednesday') or: (Day =
'Thursday')))) Then
        exitLogic := NamedElement ("Exit", "Mon-Thurs")
    Else
        Breakpoint
        COMMENT: Invalid Day
    EndIf
EndIf
Return (exitLogic)
```

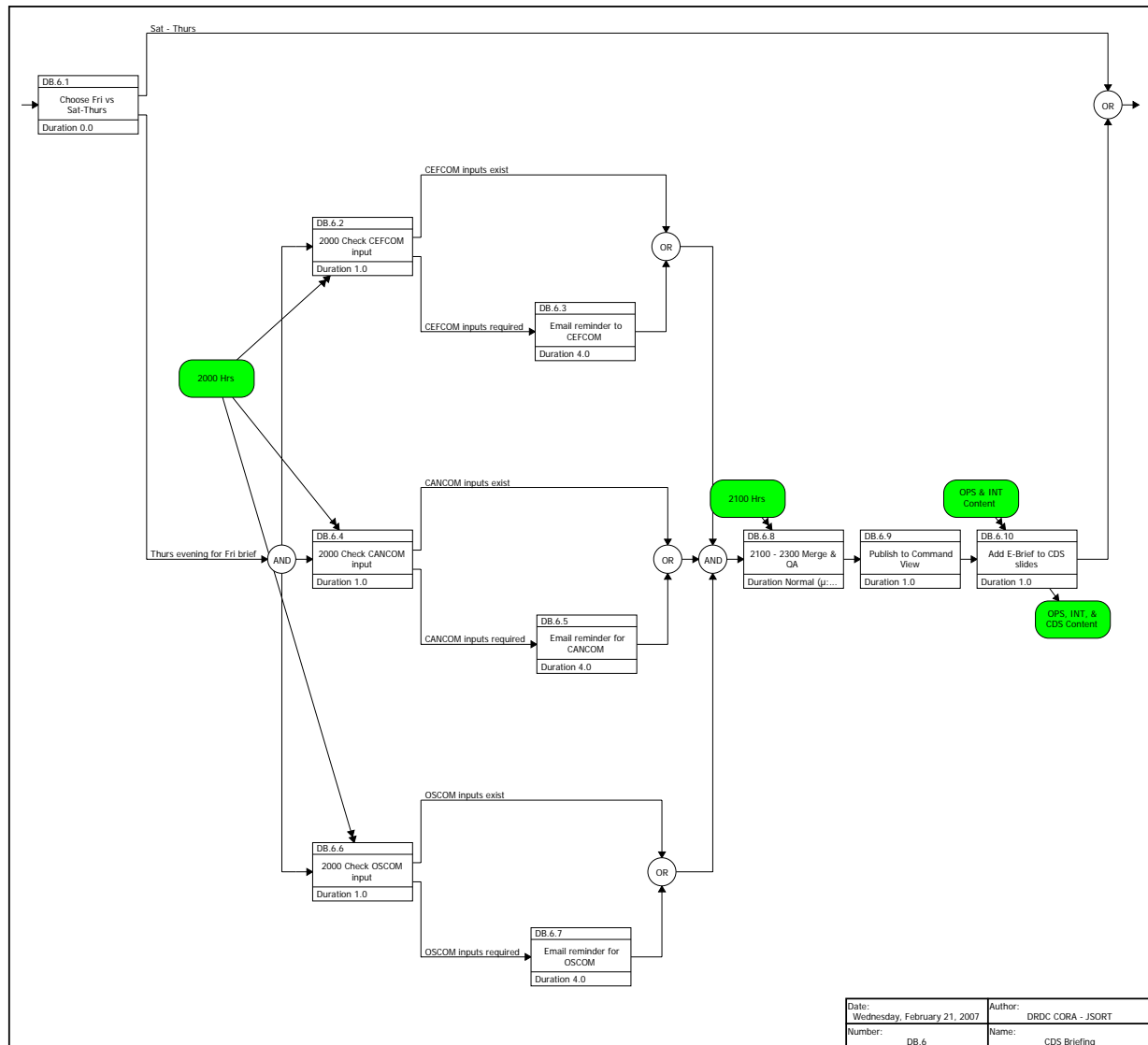



Figure 64: DB.6 CDS Briefing EFFBD

OperationalActivity Exit Logic Rules:

DB.6.1 Choose Fri vs Sat-Thurs

(Script: Untitled)

If (Day = 'Friday') Then

 exitLogic := NamedElement ("Exit", "Thurs evening for Fri brief")

Else

 exitLogic := NamedElement ("Exit", "Sat - Thurs")

EndIf

Return (exitLogic)

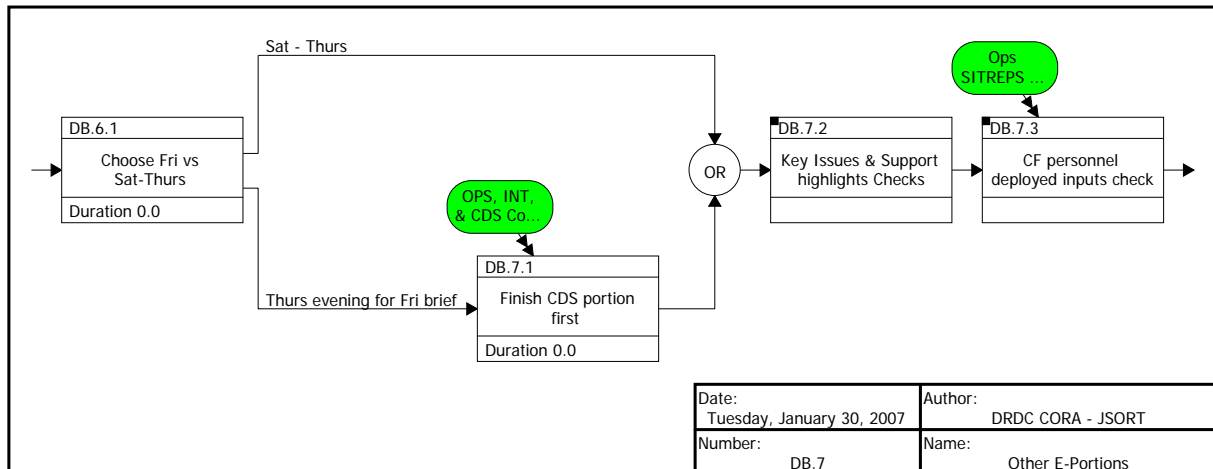


Figure 65: DB.7 Other E-Portions EFFBD

OperationalActivity Exit Logic Rules:

DB.6.1 Choose Fri vs Sat-Thurs

(Script: Untitled)

If (Day = 'Friday') Then

 exitLogic := NamedElement ("Exit", "Thurs evening for Fri brief")

Else

 exitLogic := NamedElement ("Exit", "Sat - Thurs")

EndIf

Return (exitLogic)

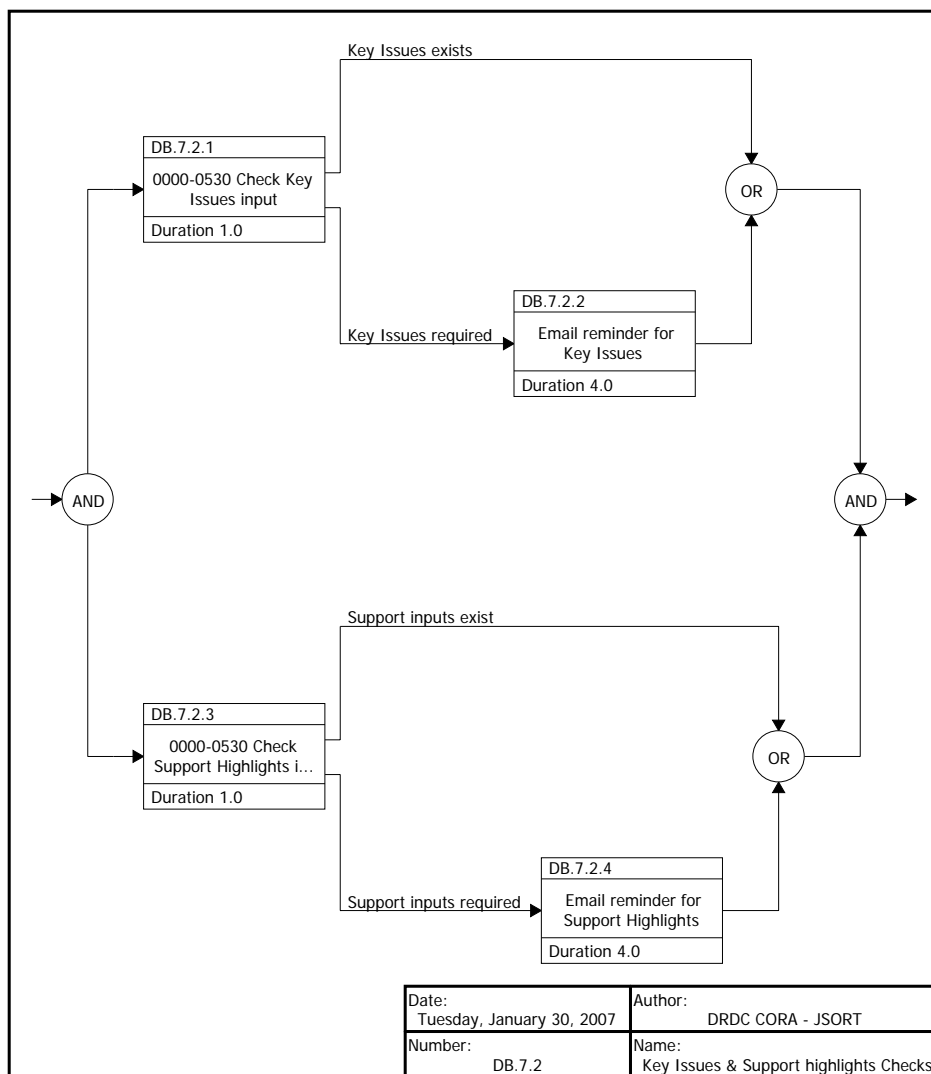


Figure 66: DB.7.2 Key Issues & Support highlights Checks EFFBD

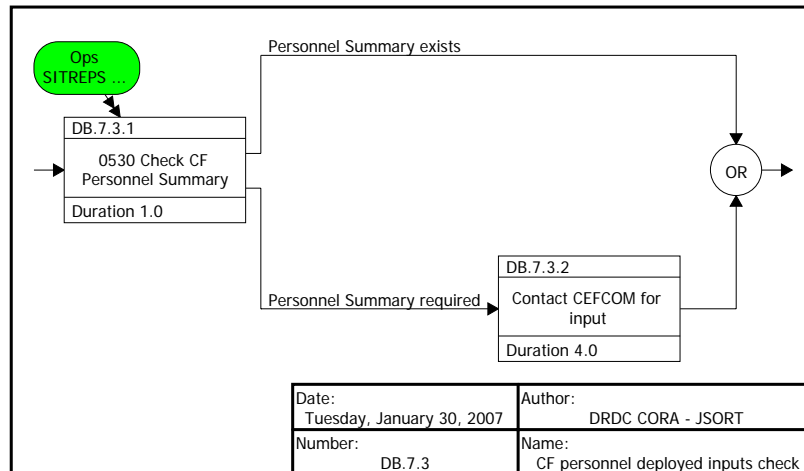


Figure 67: DB.7.3 CF personnel deployed inputs check EFFBD

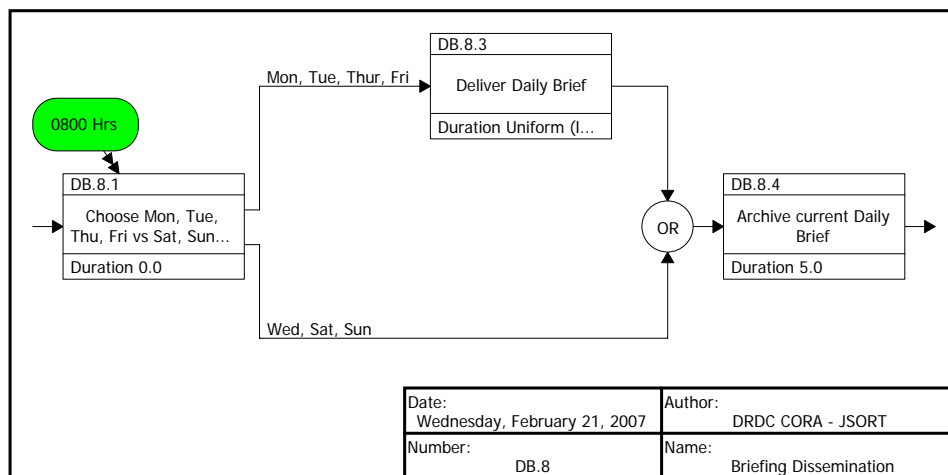


Figure 68: DB.8 Briefing Dissemination EFFBD

OperationalActivity Exit Logic Rules:

DB.8.1 Choose Mon, Tue, Thu, Fri vs Sat, Sun, Wed

(Script: Untitled)

```

If (((Day = 'Saturday') or: (Day = 'Sunday')) or: (Day = 'Wednesday')) Then
  exitLogic := NamedElement ("Exit", "Wed, Sat, Sun")
Else
  exitLogic := NamedElement ("Exit", "Mon, Tue, Thur, Fri")
EndIf
Return (exitLogic)

```

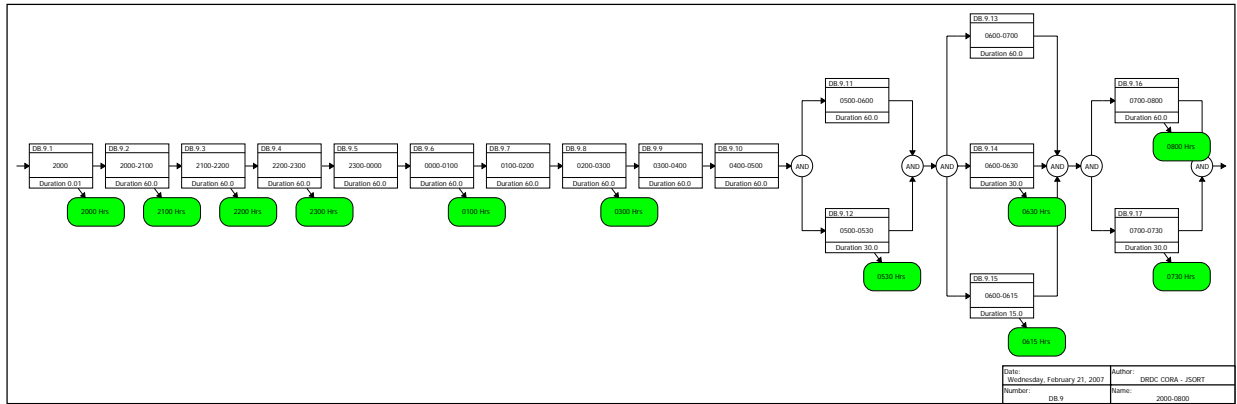


Figure 69: DB.9 2000-0800 EFFBD

Annex D Case 2 – Casualty Reporting

This Annex is included to provide all of the diagrams that make up the International Casualty Reporting process model that was described in Section 6.2.2.

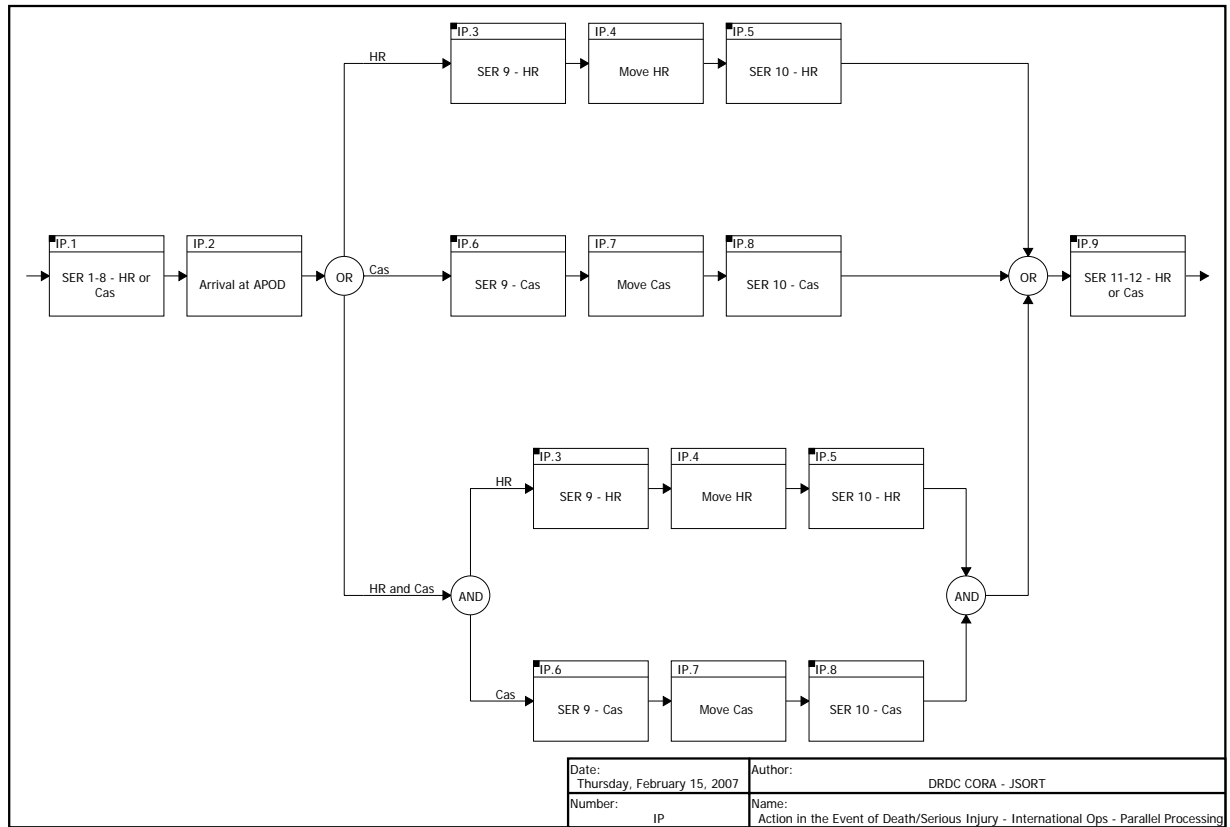


Figure 70: IP Action in the Event of Death/Serious Injury - International Ops - Parallel Processing EFFBD

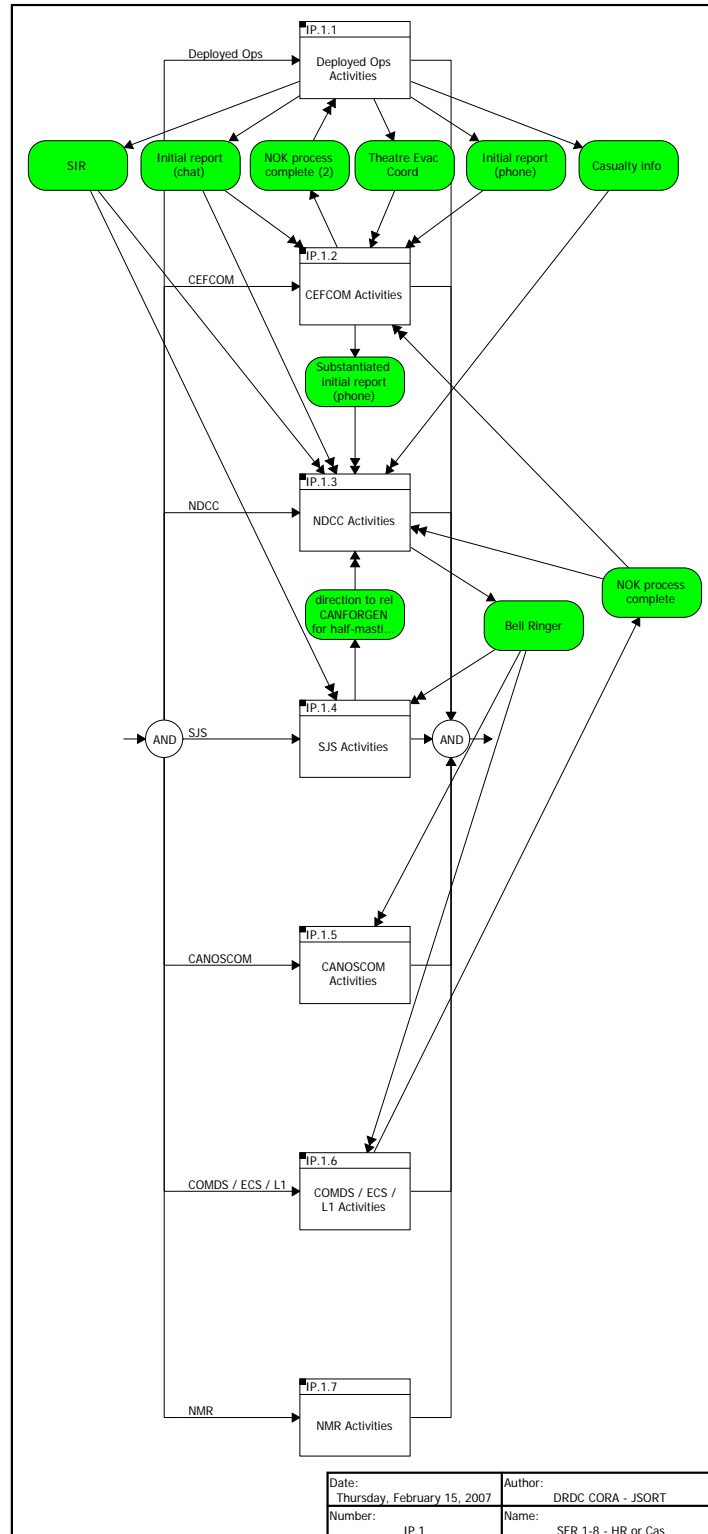


Figure 71: IP.1 SER 1-8 - HR or Cas EFFBD

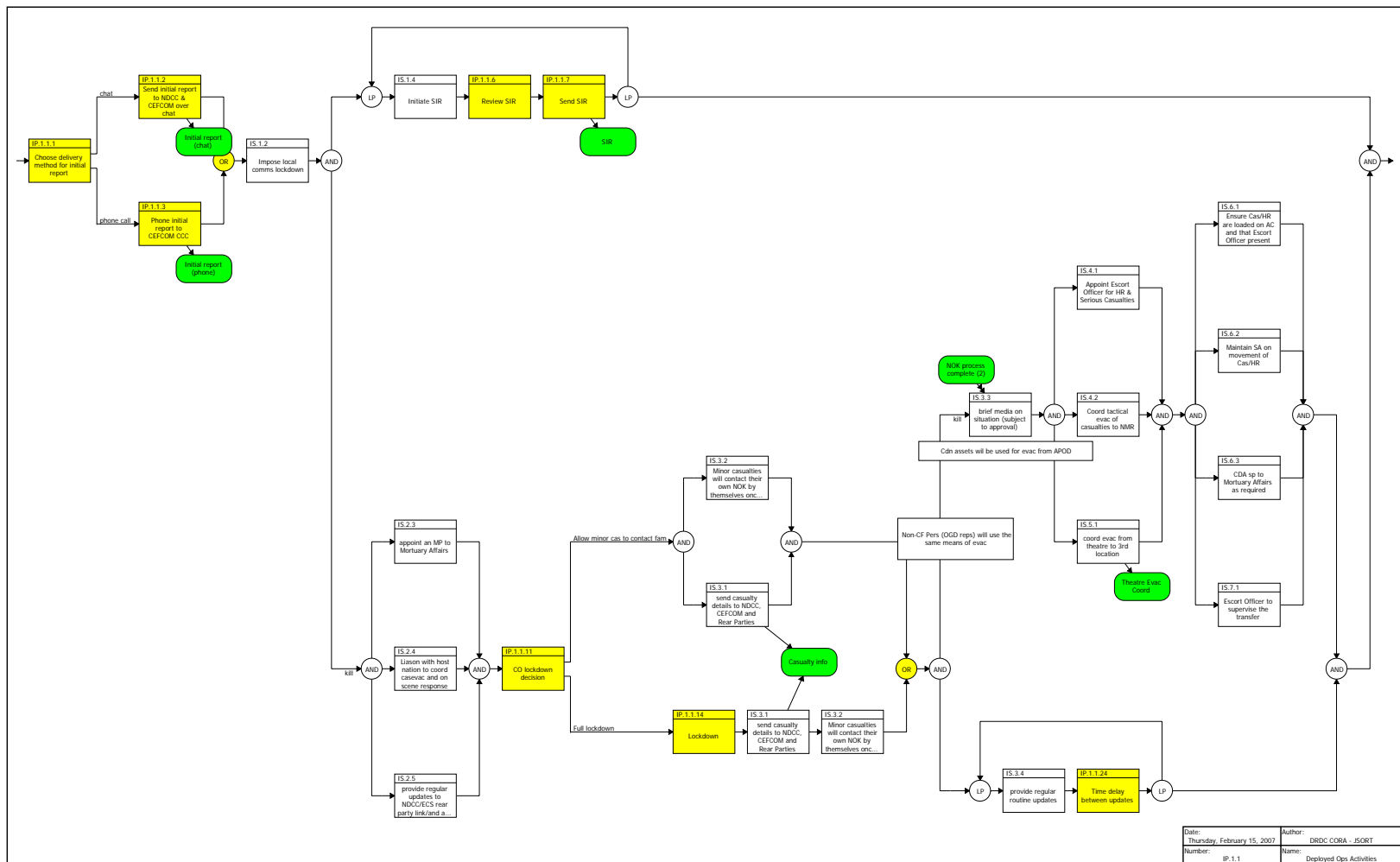
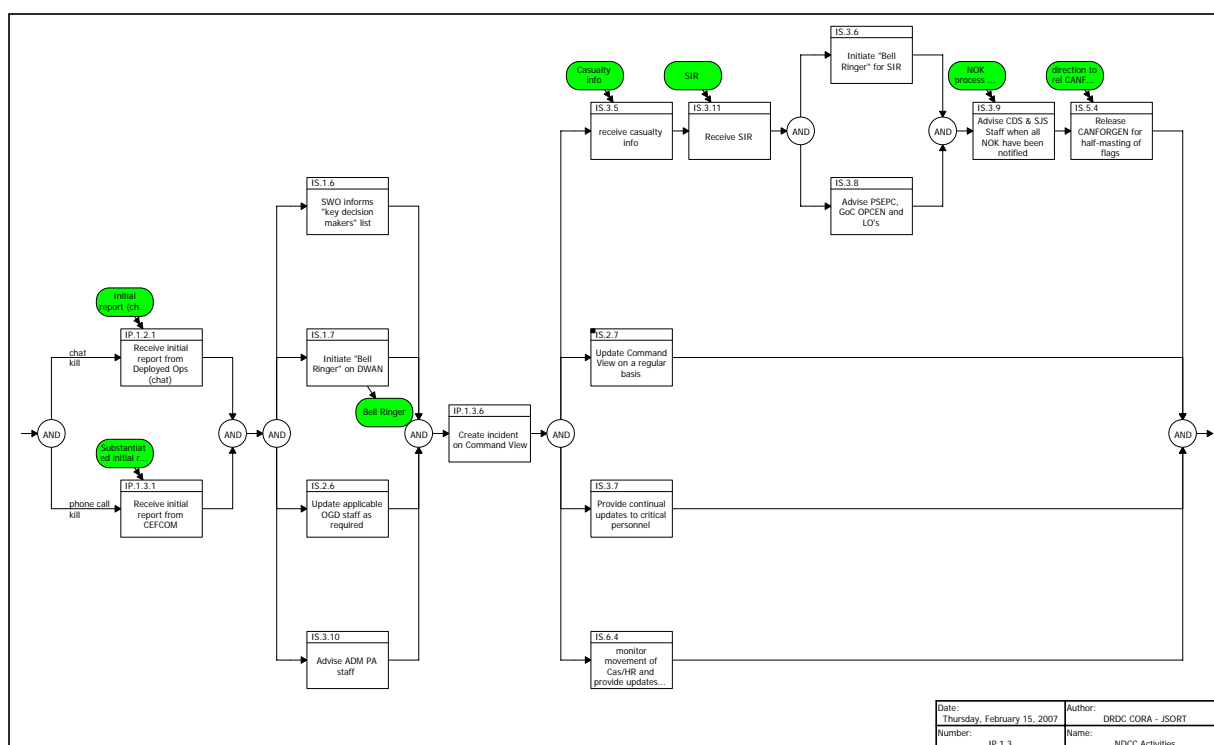
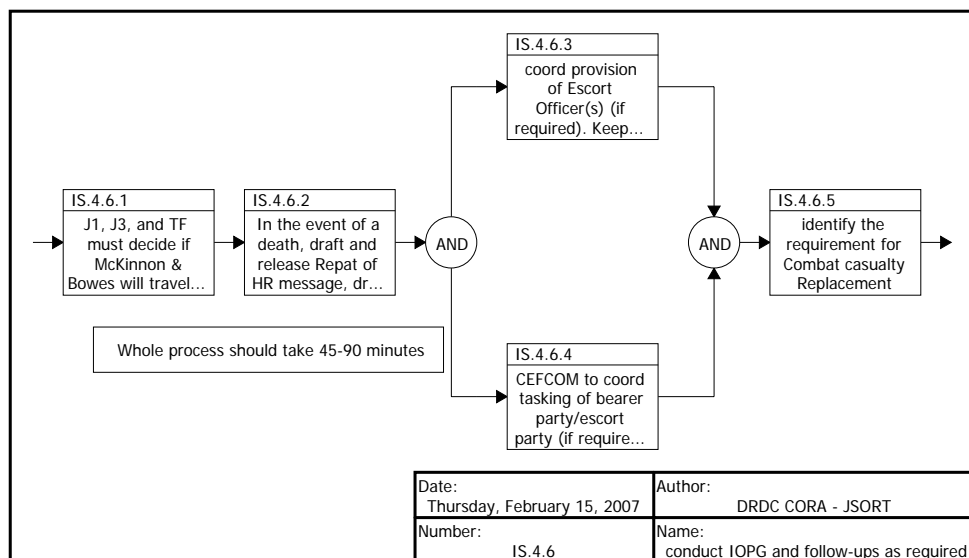


Figure 72: IP.1.1: Deployed Ops Activities EFFBD





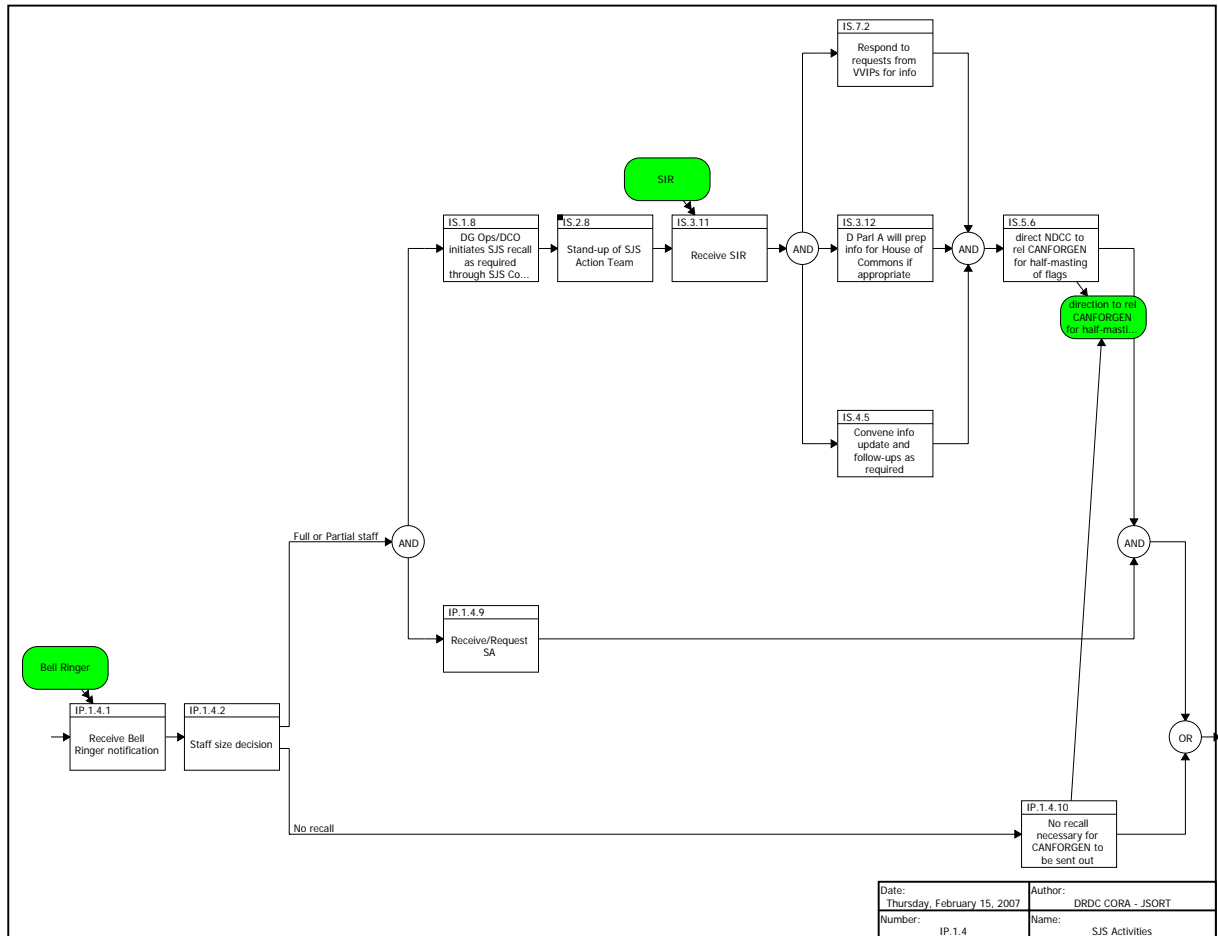


Figure 76: IP.1.4: SJS Activities EFFBD

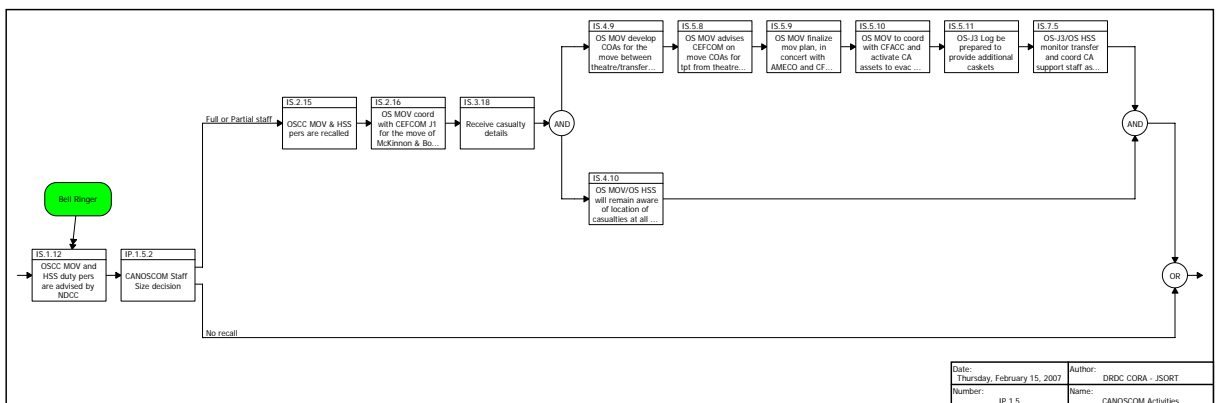


Figure 77: IP.1.5 CANOSCOM Activities EFFBD

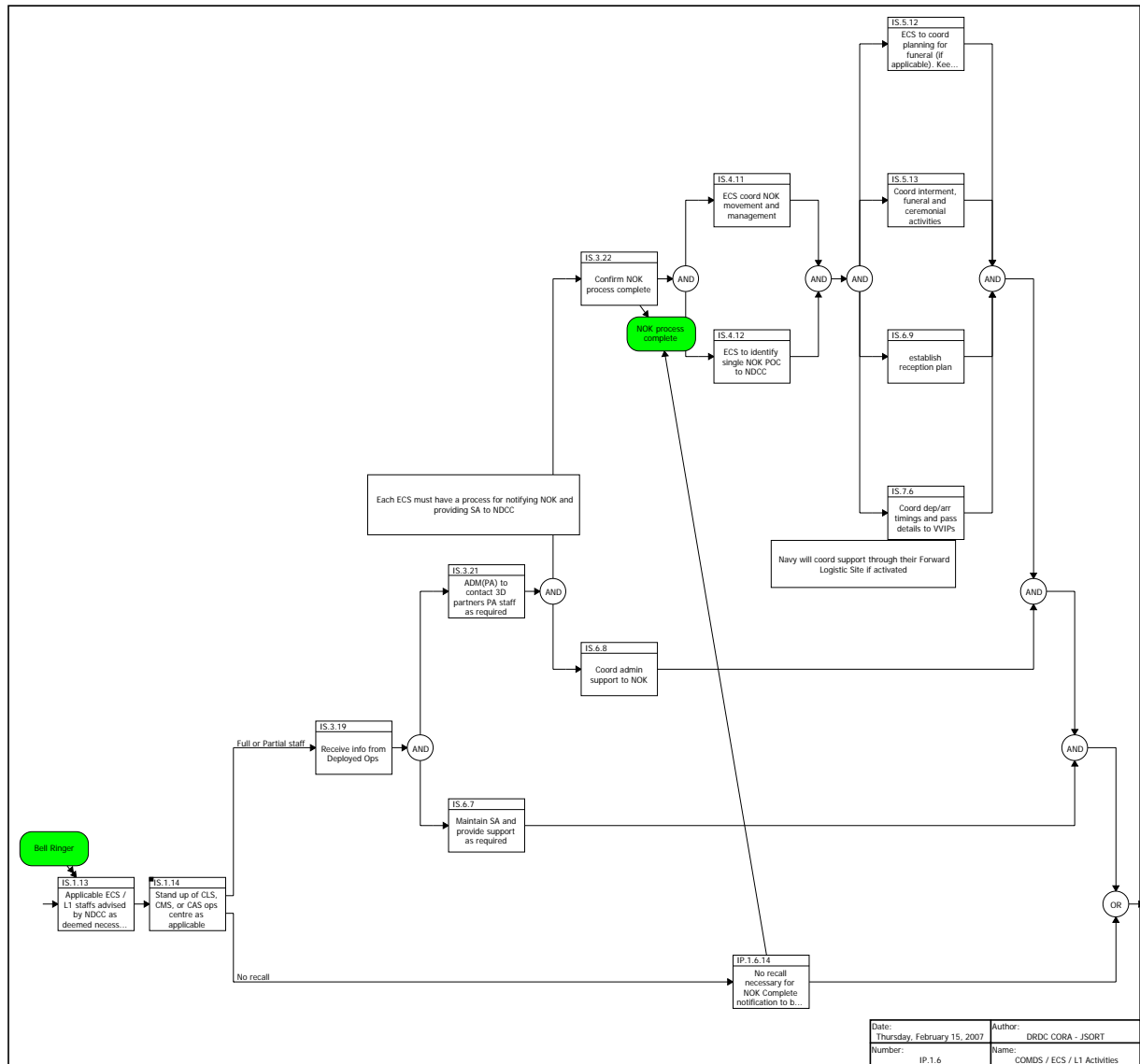


Figure 78: IP.1.6 COMDS / ECS / L1 Activities EFFBD

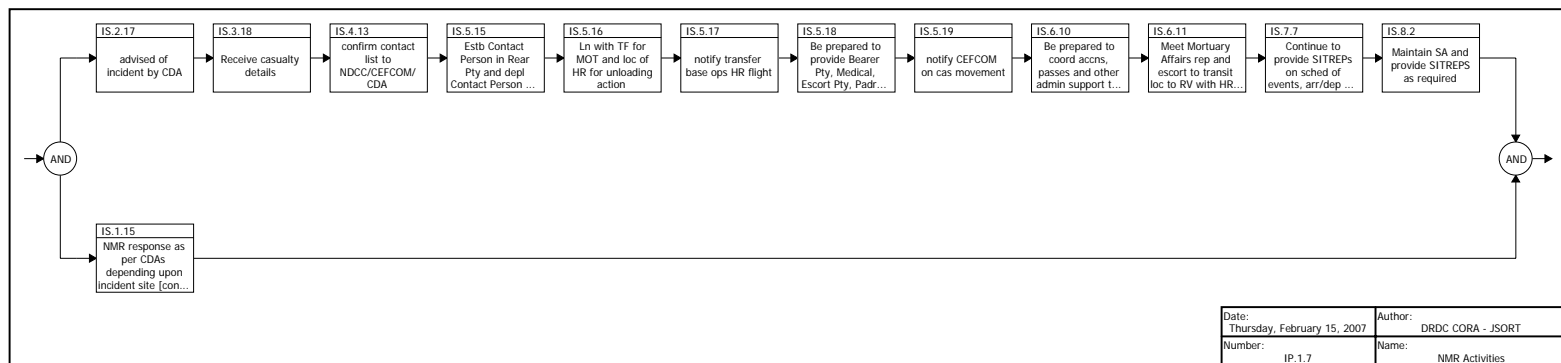


Figure 79: IP.1.7 NMR Activities EFFBD

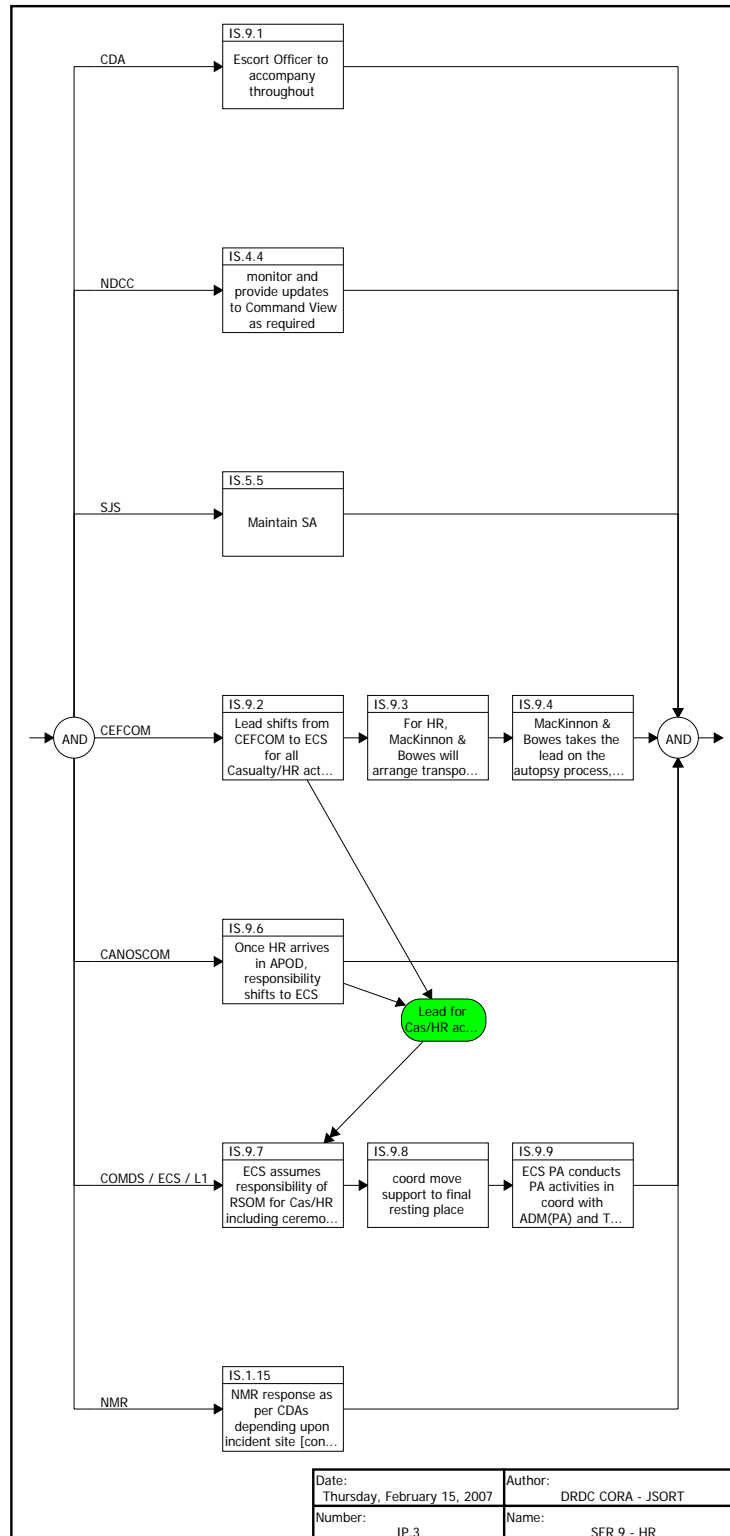


Figure 80: IP.3 SER 9 - HR EFFBD

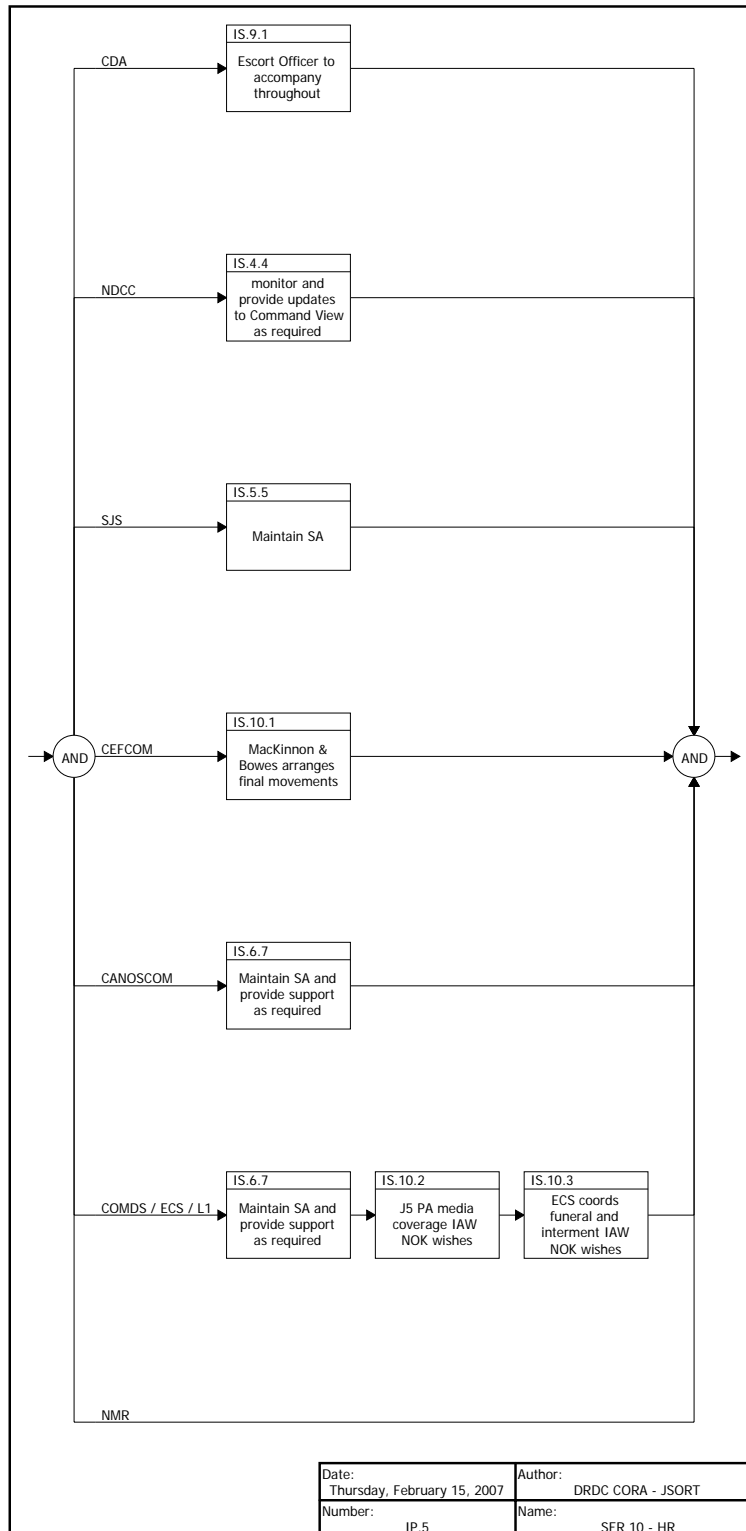


Figure 81: IP.5 SER 10 - HR EFFBD

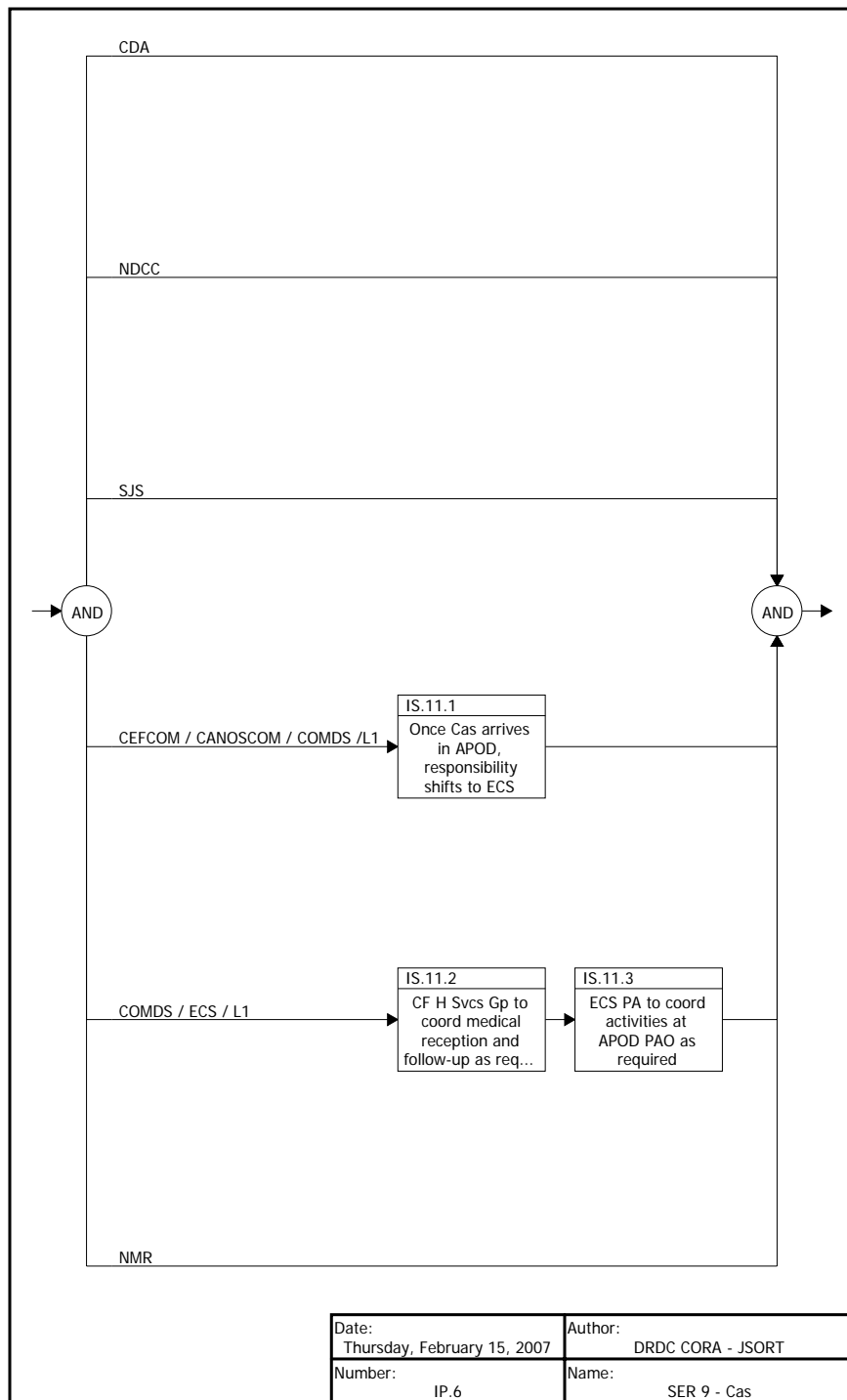


Figure 82: IP.6 SER 9 - Cas EFFBD

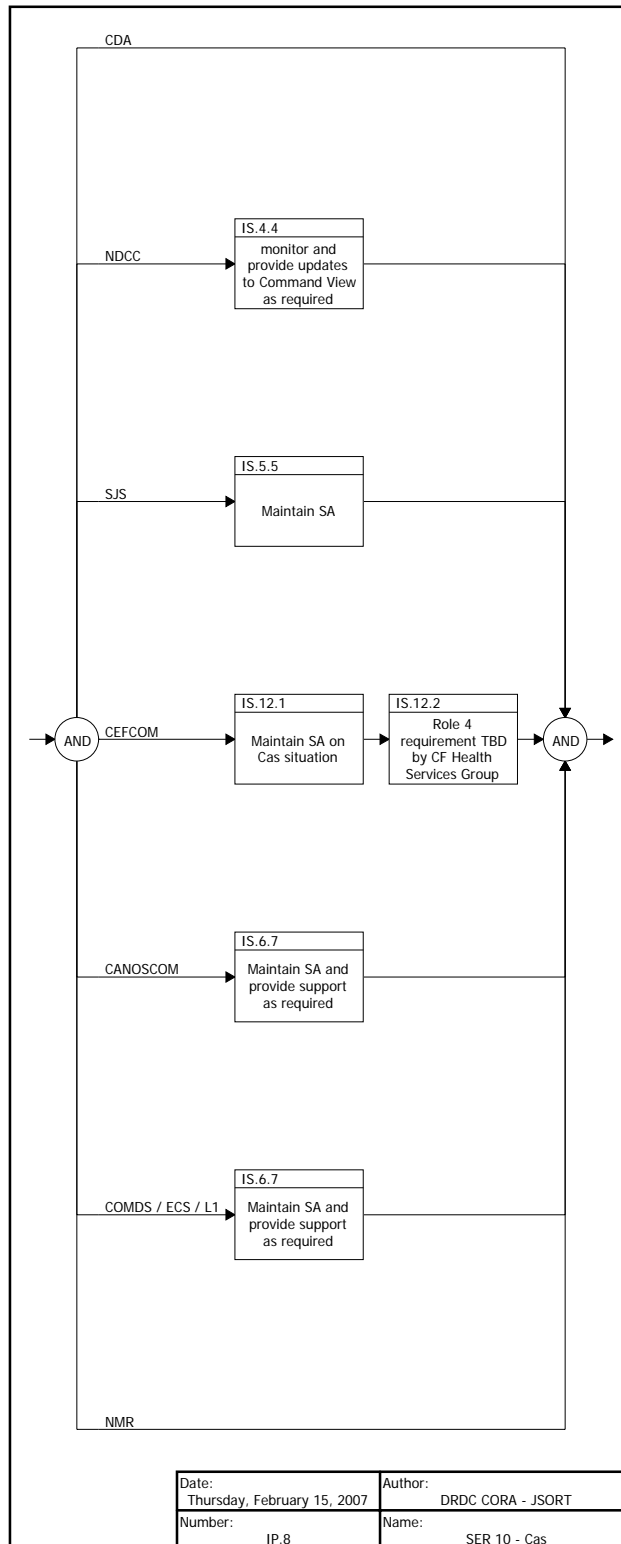


Figure 83: IP.8 SER 10 - Cas EFFBD

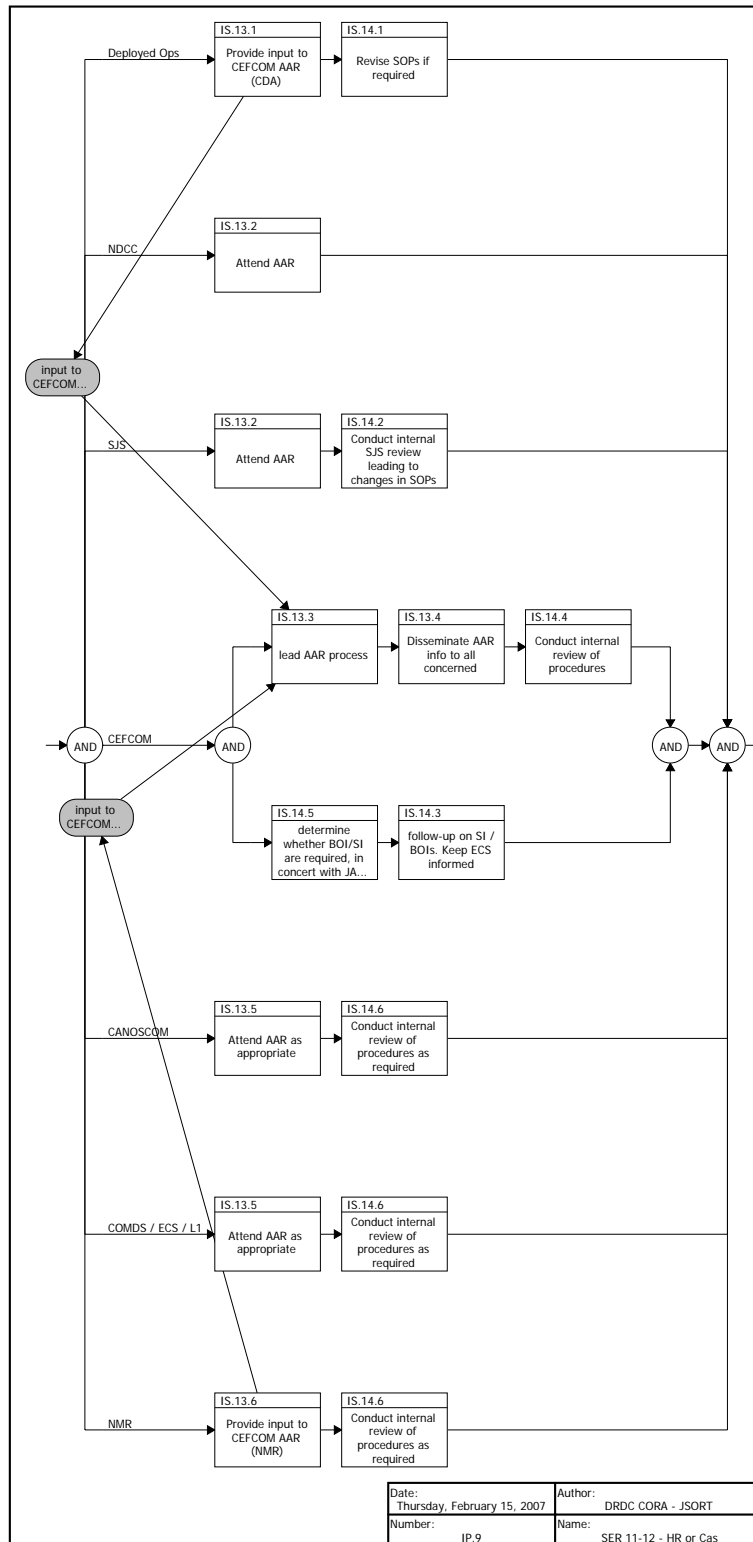


Figure 84: IP.9 SER 11-12 - HR or Cas EFFBD

Annex E Future SOP Considerations

The following SOPs have been identified by operators as good candidates for modelling in CORE and conversion to SMOFN inputs, as described in Chapter 6:

E.1 SME Proposed SOP Threads

The following activities are partially modelled through SME interviews, but must be validated:

1. RFI process for CDI and for Ops
2. OSINT/OSINF
3. JCDS21 I2 processes
4. CFEC Guardian

E.2 SJS NDCC Proposed SOPs

The following SJS NDCC SOPs have been proposed for modelling and validation:

1. Daily e-brief: NDCC version has been described and validated by SME, repeat for all Commands and link for interactions
2. Casualty/HR: Intl Ops SOP matrix has been reviewed and validated by SME, repeat for versions from Canada COM & Domestic Ops
3. Incident/Event Reporting
4. Main operational event to trigger other activities
5. Staff Action Teams (Crisis OPP)
6. Initial response (High Fidelity)
7. Follow-on activities (Lower Fidelity)
8. NORAD Responsibilities & SOPs (classified version)
9. Commander's Critical Intelligence Requirement
10. Deliberate planning
11. Other briefs

E.3 JIIFC Det proposed SOPs

The JIIFC Det has proposed the following SOPs for modelling:

1. Visits to theatre (i.e. JTF AFG)
2. J4 Log tracking
3. J1 - J9 cooperation
4. Data Transfer
5. OPCEN support to Commands

This page intentionally left blank.

List of symbols/abbreviations/acronyms/initialisms

AAR	After Action Report
APOD	Airport of Disembarkation
C2	Command and Control
Cas	Casualties
CDS	Chief of the Defence Staff
CEFCOM	Canadian Expeditionary Forces Command
CF	Canadian Forces
CFACC	Combined Forces Air Component Command
CFOCC	Canadian Forces Operations Centre Conference
Comd	Commander
CORA	Centre for Operational Research and Analysis
DND	Department of National Defence
DNDAF	DND/CF Architecture Framework
DoDAF	Department of Defense Architecture Framework
DRDC	Defence Research & Development Canada
eBrief	Electronic Briefing
ECS	Environmental Chiefs of Staff
EFFBD	Enhanced Functional Flow Block Diagram
GUI	Graphical User Interface
HR	Human Remains
IC2S	Integrated Command and Control System
JCDS21	Joint Command Decision Support for the 21 st Century
JSORT	Joint Studies Operational Research Team
LE	Loop Exit
LP	Loop
MARLANT	Maritime Forces Atlantic
MBCE	Model-Based Capability Engineering
MSOC	Marine Security Operations Centre
NAPP	National Aerospace Planning Process
NDCC	National Defence Command Centre

NLT	No Later Than
NSTR	Nothing Significant to Report
OPCEN	Operations Centre
OV	Operational View
OWO	Operations Watch Officer
SA	Situational Awareness
SITREP	Situation Report
SJS	Strategic Joint Staff
SM	State Machine
SME	Subject Matter Expert
SMOFN	State Machine of Federated Nodes
SOP	Standard Operating Procedure
SWO	Senior Watch Officer
TPED	Task, Process, Exploit, Disseminate
TPPU	Task, Post, Process, Use
VKB	Virtual Knowledge Base
VV&A	Verification, Validation, and Accreditation
NOTE	<i>Acronyms used in the diagrams of the Annexes are NOT listed because specific model content is incidental to the issues discussed in this report.</i>

Glossary

Federated

Several smaller entities operating as one large interconnected entity. We consider the operational concept of federated nodes, while the system equivalent would be a federated database [1]. It stands to reason that the physical implementation of the Repository, which is shared by the federated nodes, would be a federated database.

Job

A series of steps by one or more operators. The result of each instance of each job is a unique product. For example, if an OPCEN produces several SITREPs, the production of each one is a separate instance of a SITREP job.

Node

A logical operational entity. It is used at multiple levels of abstraction such as the different roles performed by an OPCEN (Producer, Consumer, etc.), of the OPCENs themselves, or of the different roles within the chain of command.

Post

The act of storing the new information produced from a job step. The post may occur one or more times within a job step but must occur at the end of each step.

Slack time

The amount of time that a job can be delayed and still be completed on time.

State Machine

“Any device that stores the status of something at a given time and can operate on input to change the status and/or cause an action or output to take place for any given change.” [4]

Step

Jobs are divided into a series of steps. In the SMOFN, each step can have a different skill requirement, duration, amount of utility contributed, and number of posts. There is always a post of the interim or completed product into the repository at the end of each step.

Thread

Similar to a job except that the context is different. A thread focuses on describing the series of steps performed whereas the job focus is on the output.

Distribution list

Document No.: DRDC CORA TR 2009-012

LIST PART 1: Internal Distribution by Centre

- 1 DG CORA
- 1 DDG CORA
- 1 DG DRDC Valcartier
- 1 DG DRDC Ottawa
- 1 DG DRDC Atlantic
- 1 SH Maritime
- 1 SH Land
- 1 SH Air
- 1 SH JCOR
- 2 CORA Library (1 hard copy + 1 PDF)
- 3 JSORT attn Mark Ball (2 hard copies + 1 PDF)
- 1 MARLANT N02OR
- 2 MARPAC J02OR attn Ron Funk (1 hard copy + 1 PDF)
- 1 EXORT

18 TOTAL LIST PART 1 (4 hard copies + 14 PDF)

LIST PART 2: External Distribution by DRDKIM

- 1 Library and Archives Canada
- 1 DGRDP
- 1 CFD
- 1 CDI
- 1 DJCP
- 1 CFEC
- 1 JIIFC PD
- 1 JIIFC Det
- 1 DEA
- 1 JCDS21 TD PM
- 1 MSOC PM
- 1 IC2S PM
- 1 MECSS TD PM

13 TOTAL LIST PART 2 (13 PDF)

31 TOTAL COPIES REQUIRED (4 hard copies + 27 PDF)

This page intentionally left blank.

DOCUMENT CONTROL DATA		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) DRDC Centre for Operational Research & Analysis National Defence Headquarters Ottawa, ON K1A 0K2	2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.) UNCLASSIFIED	
3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.) Executable Architecture of Net Enabled Operations: State Machine of Federated Nodes:		
4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used) M.G. Ball; R.W. Funk; R. Sorensen		
5. DATE OF PUBLICATION (Month and year of publication of document.) November 2009	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 144	6b. NO. OF REFS (Total cited in document.) 12
7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.) Technical Report		
8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.) DRDC CORA		
9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)	
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.) DRDC CORA TR 2009-012	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)	
11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.) Unlimited		
12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.) Unlimited		

13. **ABSTRACT** (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

Abstract

The Defence Research and Development Canada (DRDC) Centre for Operational Research and Analysis (CORA) is developing capability-engineering analysis tools to support the building, demonstration, and analysis of executable architectures. Our previous paper [7] described how to model workflows within an Operations Centre (OPCEN) employing a Net-Centric architecture. It used a State Machine (SM) to simulate how multiple jobs can proceed in parallel, and in contention within priorities, when operators use a Task, Post, Process, Use (TPPU) cycle to organize their work.

This paper extends the OPCEN SM model to track the interaction of work between OPCENs. The State Machine of Federated Nodes (SMOFN) engine is organized around networked functional nodes within those OPCENs that produce and consume products held in a virtual Repository. The data-driven simulation uses files to build customized job workflows and configure any combination of nodes without affecting the operational logic.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Network-Centric Architecture; Modelling and Simulation; COREsim

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale



www.drdc-rddc.gc.ca

